

University of Applied Sciences Koblenz
Department of Electrical Engineering
Prof. Dipl.-Inf. H. J. Unkelbach
University of Western Australia
School of Electrical, Electronic and Computer Engineering
Prof. Dr. rer. nat. habil. T. Bräunl

Diplomarbeit

Control for a Biped Robot with Minimal Number of Actuators

Antonio Pickel

Student # 494902

November 2002 - May 2003

Abstract

In this thesis, a control for a minimal biped robot is presented. With a height of about 40cm, this robot is quite small, but this is his advantage. Conventional robots used for scientific investigation are often quite large, placing demands on resources such as external power supply and require complex handling of both hard- and software aspects. In contrast, smaller robots are much cheaper, less complex to handle, and require fewer resources. Therefore, minirobots are investigated in different research areas, for example the biped walking field. These robots provide the opportunity to scale developed solutions to larger platforms.

The robot used for this research was the first prototype with this mechanical construction and was never before programmed and tested. Consequently, the design and the mechanical construction had to change several times during the course of the project. The mechanical changes made, took a long time, as is typical for a prototype, and thus the main focus of this thesis are the mechanical, electrical, and control engineering aspects. After the design of these parts was finished, the software part could be implemented for the system.

Acknowledgements

First, I like to thank my supervising Prof. H. Unkelbach from my home university, whose support made it possible to realize this thesis.

I also would like to thank The University of Western Australia, and specially Prof. T. Bräunl for giving me the opportunity to do my research and write my thesis in his facilities. Without his offer and assistance, this thesis would not have come into existence.

Many thanks also to the electronic and mechanical workshop of the UWA, for helping me to build and change the robot.

Last, but not least I would like to thank Jochen, Jia, Christoph, Siddharth and Norman for their advice during this thesis, and for the good times we have had working together.

Table of Contents

1. Synopsis.....	1
2. Introduction	2
2.1. Why biped robots?	2
2.2. Basics of two legged walking.....	3
2.2.1. Gait phases.....	4
2.2.2. Static and dynamic balance.....	5
2.2.3. Walking and running	7
2.3. Review and research.....	8
2.3.1. Legged machines	8
2.3.2. Actuators used for biped robots	11
2.3.3. Sensors used for bipeds.....	13
3. Mechanical design.....	14
3.1. Theory	14
3.2. Leg design and mechanism	15
3.3. Moving mass system	17
3.4. Foot design.....	21
3.5. Prototypes and actual design	22
4. Hardware and Software.....	24
4.1. The EyeBot	24
4.2. Actuators.....	25
4.3. Sensors	27
4.4. Software environment	30
5. Modelling and Simulation.....	32
5.1. Theoretical analysis of the legs	34
5.2. Experimental analysis of the model	37
5.3. Choice of a controller for close loop control.....	42
5.4. Simulation with WinFACT	44
5.4.1. Implementation of the model.....	44
5.4.2. Implementation of a controller	45

5.4.3.	Optimisation of parameters	46
5.5.	Realization of the close loop control with C++	49
6.	Static balancing control	51
6.1.	Centre of mass	51
6.2.	Sensor feedback	56
6.2.1.	Feet feedback.....	56
6.2.2.	Inclinometer feedback	57
6.2.3.	Position feedback of the moving mass	58
6.3.	Implementation of close loop control for static balance	59
7.	Dynamic balancing control	61
7.1.	Zero moment point.....	61
7.2.	Design of a close loop control for a walking gait	62
8.	Software architecture.....	63
8.1.	System components	63
8.2.	Diagram of the process structure.....	65
8.3.	Diagram of the leg control structure	67
9.	Evaluation and future work	68
10.	Conclusion	70
11.	Appendices.....	71
	Appendix A: Figure Index and sources.....	71
	Appendix B: References.....	73
	Appendix C: Faulhaber Specification Sheets	74
	Appendix D: Seika Inclinometer Data Sheets.....	77
	Appendix E: Reflective object sensor Data Sheet	80
	Appendix F: Sensor PCB and Circuit	82
	Appendix G: Dual Motor Drive Data Sheet.....	83
	Appendix H: 8 Channel serial 10 bit ADC Data Sheet	84
	Appendix I: Added C++ functions.....	88
	Declaration	92

Acronyms

- ADC: Analog Digital Converter
- API: Application Programming Interface
- CCD: Charged Coupled Device
- CIIPS: Centre for Intelligent Information Processing Systems
- COM: Centre of Mass
- DOF: Degrees of Freedom
- I/O: Input Output
- LCD: Liquid Crystal Display
- LTI: Linear Time-Invariant
- MHz: Mega Hertz
- MIT: Massachusetts Institute of Technology
- MMS: Moving Mass System
- NPCM: Normal Projection of the Centre of Mass
- PCB: Printed Circuit Board
- PID: Proportional, Integral and Derivate (controller)
- PWM: Pulse Width Modulation
- RAM: Random Access Memory
- RGB: Red, Green, Blue
- RoBIOS: Robot Basic Input Output System
- ROM: Read Only Memory
- SDK: Software Development Kit
- TPU: Timer Processing Unit
- UWA: University of Western Australia
- ZMP: Zero Moment Point

1. Synopsis

Biped robots may open a field for a new generation of machines. They may one day replace manpower in areas where dull or hazardous tasks are to be carried out, autonomously explore the deep-sea grounds or the surface of the Mars and personally assist people in their every day life. Thus, robotics may be considered as one of the prospective key technologies of the 21st century.

However, biped robots are typically complex in design, having numerous degrees of freedom (DOF). This is because of the complexity of the human walk and the desire to design bipeds that mimic human walking, or even running. Consequently, numerous motors are used to provide the robot with as much mobility as possible, which tends to make the biped heavy, expensive and difficult to build up an apt controller. Furthermore, because of the complex controller, powerful hardware is needed to calculate the algorithm in real time. For this biped, a planar leg mechanism was constructed, with each leg actuated by one DC motor. This makes it easier to control even if it has not the mobility as a complex biped robot.

In the next chapters, I will give an overview over the actual results of the research in this field so far. I will provide a brief synopsis of the human gait to point out the complete problem. In addition to this, the next point to mention is the special requirements for the mechanical construction and the used technologies.

2. Introduction

In general, the stability of walking machines decreases according to the number of legs. Consequently, at the beginning of the development the only robots that could stand upright had four or more legs, like their examples in nature. However, the biggest challenge is biped walking like that of a human being, which was improved over many centuries of evolution. Since approximately 1980, the research has been more focussed on biped walking.

2.1. Why biped robots?

The human beings and almost all on land living animals use legs for locomotion. However not many machines were built using legs for movement. The reasons therefore are the complex design and control. Nevertheless, the main advantage of waking machines is that in contrast to wheeled robots, they do not need a customized environment. They could be able to move in an environment that is only accessible by human beings. In theory not only walking but also running, jumping, climbing or even swimming could be implemented. In contrast, wheeled machines need a relative planar terrain and enough space to avoid obstacles. Bipedes use different support areas for carrying their weight and getting grip and are in the ideal case as fast and flexible as a human. Using this flexible support on the ground, a large adaptability is achieved. The legs can also be considered as an individual suspension system whereby the upper part of the body moves forward on another trajectory as the feet. Decoupling the legs from the rest of the body allows carrying payload smooth through a rough terrain. Both types of robots are designed for a specified environment: The wheeled robots are more efficient on a planar surface whereas walking machines have an advantage on all other terrains.

The operational area of robots, especially with two legs, is the natural setting of humans. The human body has, because of his anatomy, an exceptionally manoeuvrability which is perfect exploited for his locomotion. Thus, he can adapt to a new environment with minimal effort.

2.2. Basics of two legged walking

To understand the topic of the biped walking an overview of a human model will be shown. For the reason that most of the humanoid robots use the human body as paragon, it is suggestive to use the same terminology as for the human anatomy. There are three basic planes referred to as frontal (or coronal), sagittal and transversal as shown in Figure 2.1. [1]

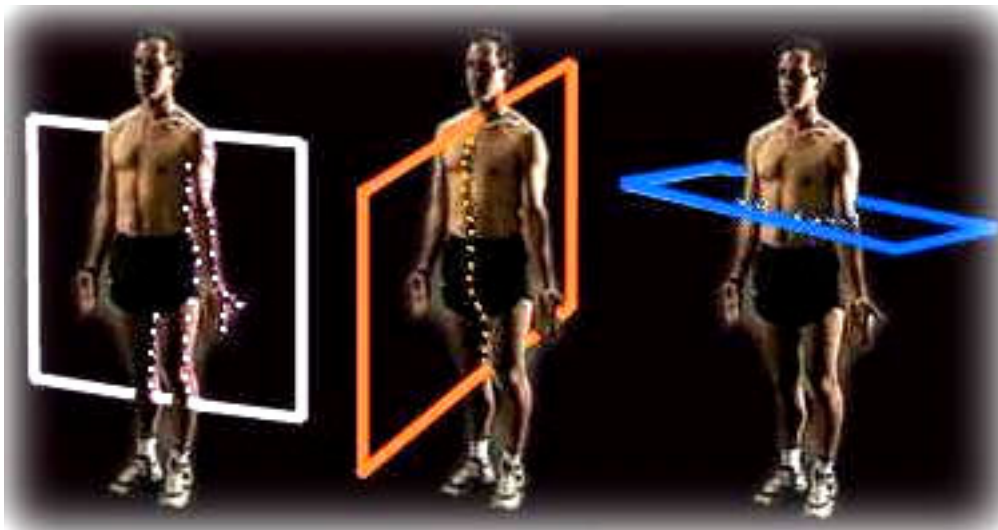


Figure 2.1: The three anatomic planes: frontal, sagittal and transversal

2.2.1. Gait phases

Walking is a cyclic movement consisting of two main phases, which alternates on both legs (see Figure 2.2 and 2.3).

During the double support phase (I), both feet are in contact with the ground. In this phase, the body has a stable position because of the wide support area on the ground. The system enters this state with the heel strike (IV) and exits it with the toe off (II) movement.

During the single support phase, only one foot is in contact with the floor. In this state, the centre of mass (COM) of the system rotates like an inverted pendulum above the contact point. Meanwhile the swing leg moves forward (III) to touch the ground again and enter the other phase.

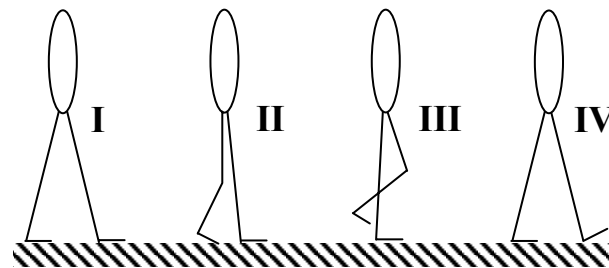


Figure 2.2: Leg position during one-half cycle

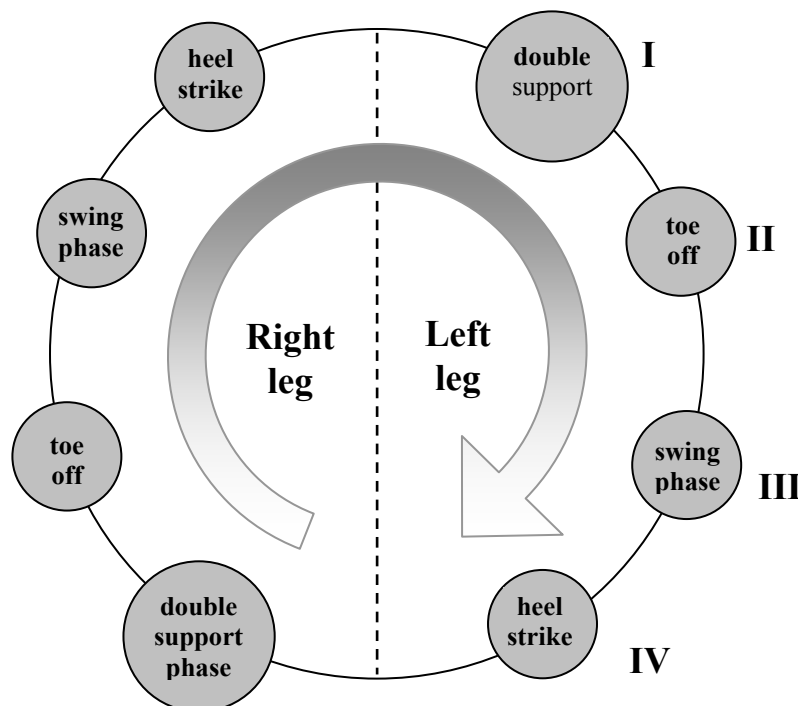


Figure 2.3: The cyclic phase rotation of biped walking

The COM of a walking human oscillates continuously four centimetres up and down and simultaneously left and right. The point is located in the hip (see Figure 2.4). For the development of a walking gait, there are three important parameters (see Figure 2.4): the step length (d), the step height (f) and the step period [2]. The controller can use these parameters to stabilize the gait. They also affect directly the speed and the position of the entire system.

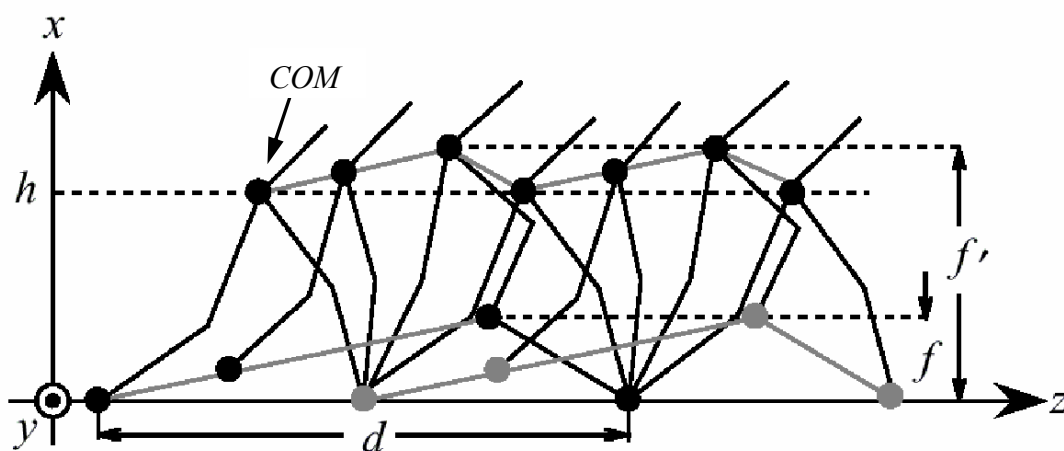


Figure 2.4: Side view of the movement of the legs during a walking gait

2.2.2. Static and dynamic balance

Walking can be divided into two main groups: walking with static balance and walking with dynamic balance.

During a static walk the normal projection of the centre of mass (NPCM) always stays in between the boundaries defined by the feet. If both feet are on the ground, the NPCM has to be within the polygon determined by the outer corners of the biped feet. If only one foot is in contact with the ground, the NPCM has to be within the area of this foot. While the movement is slow enough, the system dynamics can be ignored. Static walking assumes that if the system's motion is stopped at any time, it will stay in a stable position indefinitely. However, the speed achieved using static walk is not that high and the efficiency is far away from the human walking speed.

To introduce the semi-dynamic and the dynamic walk, another terminus has to be explained. The Zero Moment Point (ZMP) is the point on the ground where the sum of all moments on the robot is equal to zero. During a dynamic movement, normally this point has to be within the boundaries of the feet exactly as the NPCM during a static movement [3].

The semi-dynamic walk is similar to the static walk except short periods of time, in which the whole system tends to be unstable and the ZMP can exceed the stability region provided by the feet. The body may be falling during this part of the gait, and unless the feet are positioned correctly, it could fall to the ground.

A dynamically stable biped, by comparison is one that moves through unstable positions in its walking gait, and needs to intelligently adjust and plan its movements to remain stable at any given time. In this walking gait, the ZMP can also move outside the supported region for a finite amount of time, but is generally in constant movement, thus the feet are in continuous motion.

Human being and animals rarely use static walk. To achieve better results with respect to speed and efficiency some kind of controlled instability must be introduced to become more similar to the paragon of nature.

2.2.3. Walking and running

The locomotion for bipeds can be divided into two main groups, walking and running. The actual research on this field concentrates principally on walking. The two forms have very different characteristics, according to the movement. However, the main difference is the contact of the feet with the ground. During walking at least, one foot is on the ground at all times, and during the double support phase even two. Whereas while running only one foot touches the ground simultaneously. Furthermore there are only short period in which the foot has ground contact followed by a long time span with ballistic movement.

Running has a few advantages. The most important advantages are the higher velocity and the high efficiency. Running allows an elastic energy recovery during the jump phase. Therefore, a running robot must have an elastic mechanism to absorb the kinetic energy and give it back in the right moment.

2.3. Review and research

2.3.1. Legged machines

“Only if a robot is able to move free in our environment, he will one day be able to be a real help for human being and carry the name humanoid robot”. With this sentence Honda’s robot Asimo (Advanced Step in Innovative Mobility) was presented in 2001 to the world (see Figure 2.5). He was the improved successor of Honda’s P2 and P3 which were build in the lately 90’s. With a weight of 43kg, he is much lighter than his predecessors are and that is one of the most important advantages. This robot is the result of about 14 years of research and approximately 10 prototypes. ASIMO can walk continuously while changing directions, taking every step smooth and natural. He also has the ability to climb up and down a flight of stairs [4].

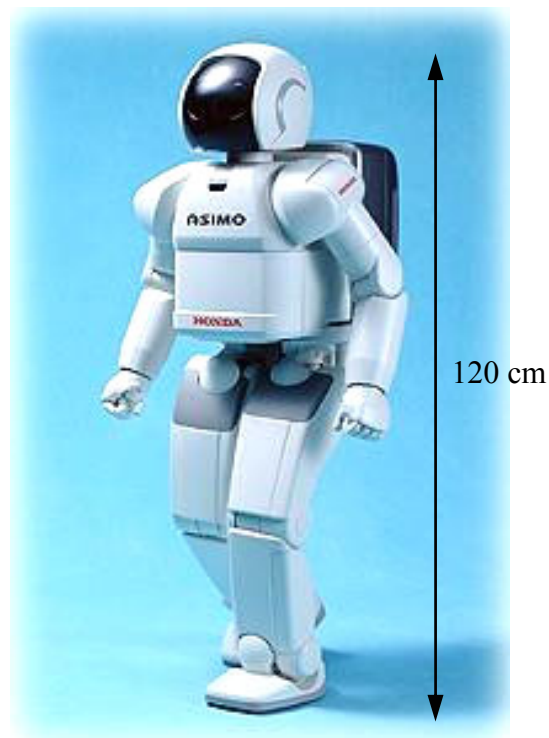


Figure 2.5: The humanoid robot ASIMO

The research of legged machines begun in the early 80’s and the Massachusetts Institute of Technology (MIT) Leg Laboratory was one of the first in developing a variety of walking, running or jumping machines.

The first Leg Lab robot was the Planar One-Leg Hopper (see Figure 2.6) [5]. It had just one leg with a small foot. It was designed to explore active balance and dynamic stability in legged locomotion. It was controlled with a simple three-part algorithm.

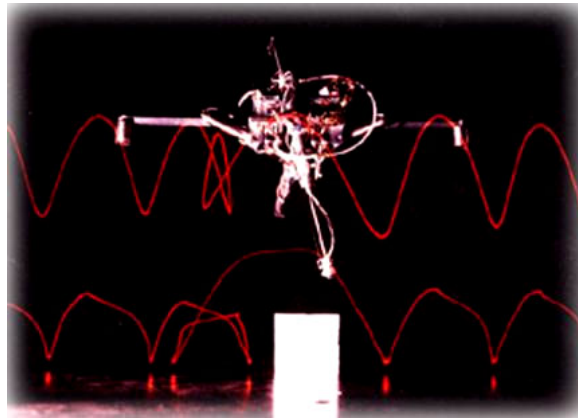


Figure 2.6: Planar One-Leg Hopper from the MIT (Raibert, M. H. 1980-82)

Following machines were built to show that actively balanced dynamic locomotion could be accomplished with simple control algorithms. The 3D One-Leg Hopper (see Figure 2.7), for example hopped in place, travelled at a specified rate, followed simple paths, and maintained balance when disturbed.

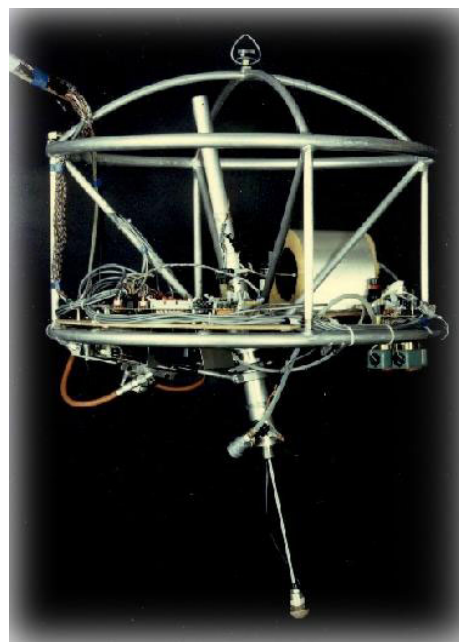


Figure 2.7: 3D One-Leg Hopper from the MIT (Raibert, M. H. 1983-84)

The research group around M. Raibert at the MIT showed that a one legged robot could stay upright with active balance, using separate control algorithm for hopping and forward movement. Furthermore, he showed that the jumping-technology could easily be upgraded to a biped running.

At the University of Western Australia (UWA) at the department of Electrical, Electronic and Computer Engineering the research group around T. Bräunl had a breakthrough in 1998 with the design of robot called “Johnny Walker” (see Figure 2.8, left) [6]. The robot used nine servos and an onboard 32-bit Controller. Each leg had 4 degrees of freedom (DOF) and with this robot, an investigation of different dynamic walking gaits was possible.



Figure 2.8: Walking bipeds Johnny walker, Jack Daniels and Andy

The main problem encountered on this first prototype was the heavy frame. The consequence was a heavy robot, which required high torques to move the legs. This problem was eliminated on another prototype called Andy (see Figure 2.8, right). The whole design was revised and for better liberty of action, the robot was provided with five DOF per leg. The result was a much lighter and flexible robot. Actual research is now taking place on this walking machine with promising results.

2.3.2. Actuators used for biped robots

Walking machines need actuators for leg movement. Usually the actuators provide a torque to move a joint. This is the cheapest and easiest way this can be done is by using a simple DC motor. For the dynamic and controller aspects, this is the simplest way to realize actuation, since the current is directly proportional to the delivered torque of the motor. On the other hand, configuring a single joint with numerous motors providing it with several DOF could be very complicated.

Given that a single joint of a human being cannot move more than a maximum angle of 300° , the idea of using servos as actuators is obvious. A servo can provide high torques despite having a small size. In addition, the positioning is very accurate being able to reach every angle exactly and to hold this position regardless which torque is acting on it. The power consumption of such an actuator is nearly the same as for a DC motor. Nevertheless, the maximum speed of a servo is limited by the gearbox and the speed of the motor inside and therefore they are not as fast as a motor.

Both actuators can provide a rotary DOF, but for translatory movement without any gearbox or crank, hydraulic or pneumatic actuators are the better choice. These actuators like pneumatic cylinder were used by the MIT for their robots for locomotion.

Besides all these actuators, there is research on specially designed actuators for this application field, which are similar to human muscles. The muscles in a human body exert a force in a certain direction while being contracted. Joints are often controlled by contraction of various muscles at the same time. The “Air Muscle” (see Figure 2.9) [7] is an attempt to use this technology for mechanical systems. The Air Muscle consists of a rubber tube covered in tough plastic netting, which shortens in length like a human muscle when inflated with compressed air at low pressure. It has a very high power-to-weight ratio, as the air-muscle itself has a weight of only 10g. This makes it especially useful for weight-critical applications. Furthermore, they have an immediate response, so the movement will be very smooth and natural. They also can be operated when twisted axially, bent round a corner, and need no precise aligning. The disadvantage of such a system is the need of additional devices such as control valves or pressure sensor gauges and of course, the compressed air needed to operate the muscles. If all this must be placed

on an autonomous robot, the robot will consequently be very heavy. Another problem is the difficult control of such a system. The more DOF the system has the more complex the control for it will be. Beside these aspects, this system comes as close to the human muscle as any other system.



Figure 2.9: Different types of air-muscles

2.3.3. Sensors used for bipeds

In order to achieve a dynamic balance for a walking robot, the use of sensors is inevitable. For a closed loop control, it is indispensable to obtain feedback from the actual joint position and alterations of the COM. Simplifying, knowing the position of the different joints, the acceleration and the inclination of the different parts of the system, the COM and even the ZMP can be calculated.

A variety of different sensors (see Figure 3.1) is used in a biped robot depending on the desired feedback value. An inclinometer (a) measures the angle of the system attached to it. The gyroscope (b) is able to deliver a change of orientation of the system. Accelerometers (c) are used when the acceleration of the system is needed as feedback.

Most bipeds also use micro switches attached at the feet to have a feedback when the foot is on the ground. Using several pressure sensors (d) distributed on foot surface, not only the ground contact of the foot is known, but also enough data is available to calculate the COM.

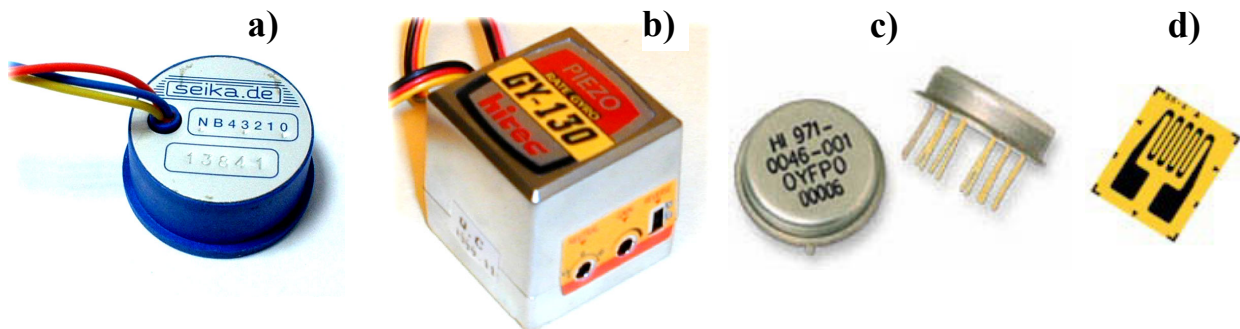


Figure 3.1: Different type of sensors: Inclinometer a), Gyroscope b), Accelerometer c), Pressure Sensor d)

From the previous research, it was found that most of the robots are increasing in complexity and therefore control of the whole system would be more difficult to implement. Therefore, a simple biped leg mechanism was designed at the School of Mechanical Engineering at UWA, using just one actuator per leg. Because of this, the whole construction is much simpler than any other system and it is easier to implement walking control using just a few sensors as feedback.

3. Mechanical design

3.1. Theory

The mechanical design of this biped robot was made by a final year student of mechanical engineering at the UWA as a bachelor thesis. The student designed the construction according to calculations made, using statics and dynamics to obtain a model of the system [8].

For the feet model, a two-dimensional system was enough, so the forces could be broken up into their rectangular components. The torque was calculated, knowing the force acting on a determinate point. The following universal equations were adapted to make the different calculations:

$$\begin{aligned} \text{Force:} \quad F_x &= F \cdot \cos \theta & F_y &= F \cdot \sin \theta \\ \theta &= \arctan\left(\frac{F_y}{F_x}\right) & F &= \sqrt{F_x^2 + F_y^2} \end{aligned}$$

$$\text{Torque:} \quad M = F \cdot d$$

The sum of all forces and torques acting on the system must be zero at every time to keep in equilibrium:

$$\sum F = 0 \quad \sum M = 0$$

The equations were applied to this problem and the system was modelled using Matlab to calculate the different matrices.

The mobility of the system was calculated using the Grübler/Kutzbach Mobility equation for the planar case and for up to two DOF:

$$Mob = 3 \cdot (n - 1) - 2f_1 - f_2 \quad \text{equation 3.1}$$

- n = Number of elements which can rotate with respect to the base
- f₁ = Number of joint with 1 DOF
- f₂ = Number of joint with 2 DOF

3.2. Leg design and mechanism

The different dimensions of the parts concerning the leg mechanism had been calculated and modelled in Matlab. Figure 3.2 shows a schematic illustration of the leg movement.

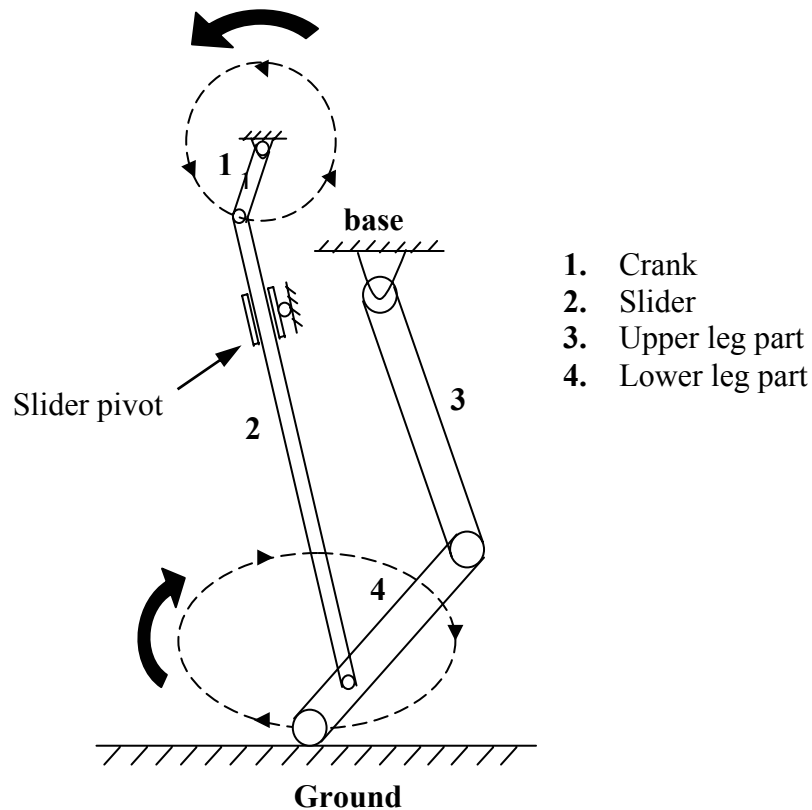


Figure 3.2: Schematic illustration of the leg mechanism

This mechanism produces a leg motion using bars, sliders and a crank to provide an active hip, knee and ankle joints. If link 1 rotates at a constant speed in an anti-clockwise direction, the slider, which is fixed with a slider pivot close to the middle of the slider, can force link 3 and 4 to move in a motion similar to a stepping foot.

The total number of moving elements is six: two bars, the crank, the slider, the foot and the slider pivot. While the number of joint with one DOF is seven, the number of joints with two DOF is zero. Inserting these values in the equation of Grübler/Kutzbach (Equation 3.1), the resulting total number of DOF per leg is one.

A kinematic analysis of the model was performed to estimate the walking path of the biped. The final dimensions and positions of each part could be optimized using this model.

After choosing the materials for each part, weight estimation could be done for each leg. Furthermore, the weight of the rest of the system was estimated, to calculate the maximum required torque.

The final design of the leg mechanism is shown in Figure 3.3 (here with the curved foot prototype). To keep the weight down, the material for most parts was Perspex (density: 1.18g/cm^3). Some components were chosen to be metallic for reasons of stiffness. Although, using aluminium, which has a density of 2.71g/cm^3 , the weight gain was tolerable. Because humans have a weight distribution of around 32% in the legs, the robots mass distribution comes close to this, having about 30% of the total mass in the legs.

The joints for this mechanism had to be manufactured carefully to minimise friction forces while moving. However, any undesired movement from the joint could complicate the whole control for the robot. Therefore, the aluminium joints in the leg were made with accuracy and inserted within the holes of each Perspex part. To fix this joint and prevent any lateral movement, screws were attached to the insert from either side. To lubricate the inserts, plastic grease was applied.



Figure 3.3: Leg mechanism

3.3. Moving mass system

During a biped walk, at least one leg is moved at a time in a pattern to produce a motion. This pattern modifies the relative position of the links to one another and thus the mass distribution of the whole system. This means that during a walking gait the COM is in constant motion and changes its position through unstable position if it is not compensated. The theory of the balancing of this robot is, having an inertial mass, which can act as a counterweight. The COM could be shifted in such a way, that any unstable position is followed by a stable position, predicting the trajectory of the system and moving the mass before it happens. The role of this mass is to provide some inherent stability in the robot due his inertia and to generate counter-moments by moving when required. The moving mass was proposed on the top of the system because of the higher the inertial mass is above the ground, the larger the change of the COM while moving it to its limits.

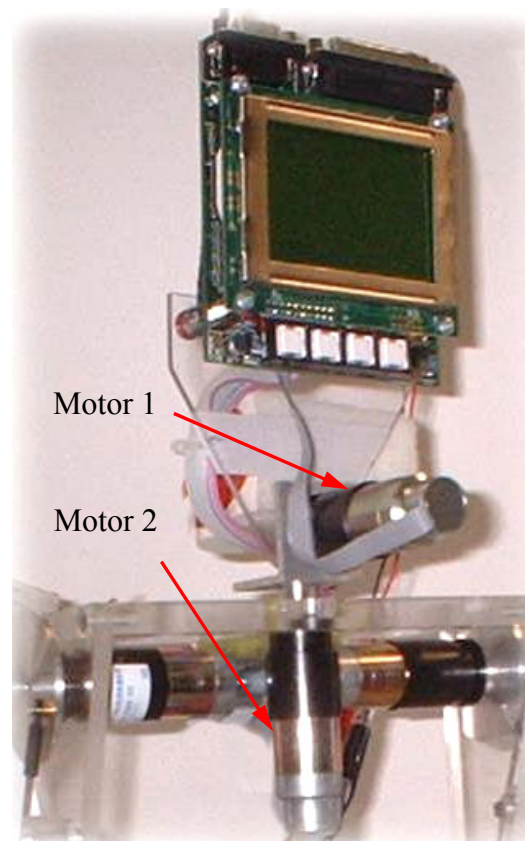


Figure 3.4: First prototype of the moving mass system

Different prototypes of this moving mass system were created in order to improve both stability and control of that part of the system. In the first prototype, (see Figure 3.4) the moving mass consists of the controller and the battery pack itself. Two DC motors were used to move the mass to a desired position. Thus, this arrangement had two DOF; One to rotate and the other to roll the mass. This system had a big disadvantage because the two movements were not pan-tilt independent and it would have been difficult to implement software control for such an arrangement.

A schematic illustration of the second prototype is shown in Figure 3.5. The mass was also actuated by two DC motors and with this arrangement, the two DOF were pan-tilt independent. However, using this arrangement, the motors had to provide a constant torque on the weight, to keep the robot upright. The mass could be modelled as a kind of a two axis inverted pendulum, and therefore the close loop control algorithm would be quite extensive. The computation time would take to long, because not only the mass had to be kept upright but also it had to be moved to certain points in real-time, to balance the robot. The fact that this system was not linear, because the trajectory of movement in each DOF was a semicircle, also aggravates the real time computation. Furthermore, the power consumption of this system was unreasonable, having two motors, which had to provide continuously a high torque, and reaching the limits of one movement, the torque needed was even higher.

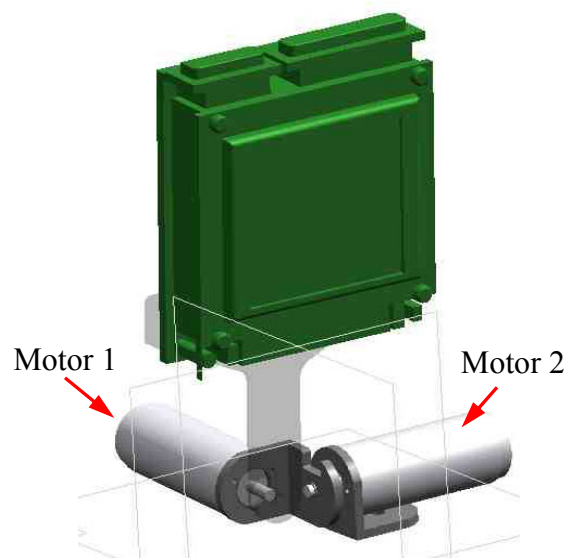


Figure 3.5: Schematic illustration of the second prototype of the moving mass system

For the next prototype, the mass itself was changed. The controller was no longer part of the moving mass system because it would cause problems not only reading the display or using the buttons when the robot is in motion, but also the connections between the different parts and the controller would be affected. Therefore, the controller was fixed on the front plate of the robot (see chapter 3.5). After calculating the COM and the changes of it needed (see chapter 6.1 for further details), to keep the robot in a stable position, the moved mass was changed in position and objects. The mass now consisted of the motor, the battery pack and different mechanical parts needed for the movement. With this arrangement, we lose one DOF, but at this stage, it was better to see how the system reacts with this mass and than add another DOF for the moving mass. As shown in Figure 3.6, the system was now linear to the axis in which it can be moved. This would be greatly advantageous. In addition, the required torque could be reduced, because no mass needed to be raised. The motor is fitted in a sliding bed (see Figure 3.7), which uses three ball bearings for a smooth frictionless movement through a notch on the hip plate of the robot. A pinion gear is fixed on the axle of the motor and in order to move the sliding bed, a rack is mounted on the upper part of the robot. The movement of the sliding bed is confined by two screws, which act like a limit stop for it. The actual design of the counterweight is shown in Figure 3.8. The maximal mass displacement is about $\pm 91\text{mm}$ from the centre and the weight of the counterweight is about 360g. The details of this mass displacement are described in Chapter 6.1.

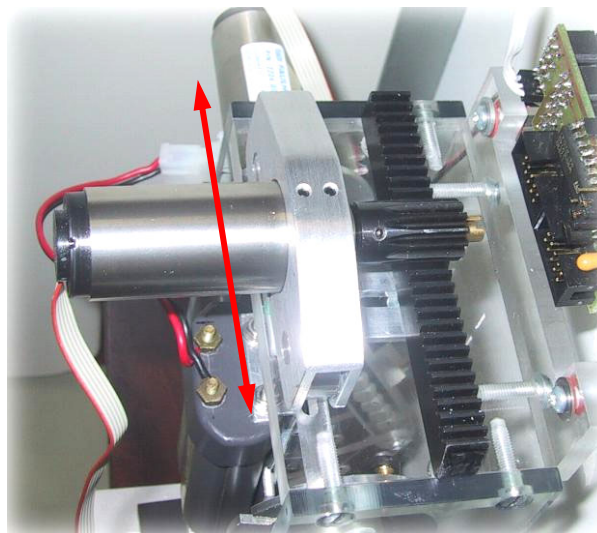


Figure 3.6: Moving mass system of the third prototype

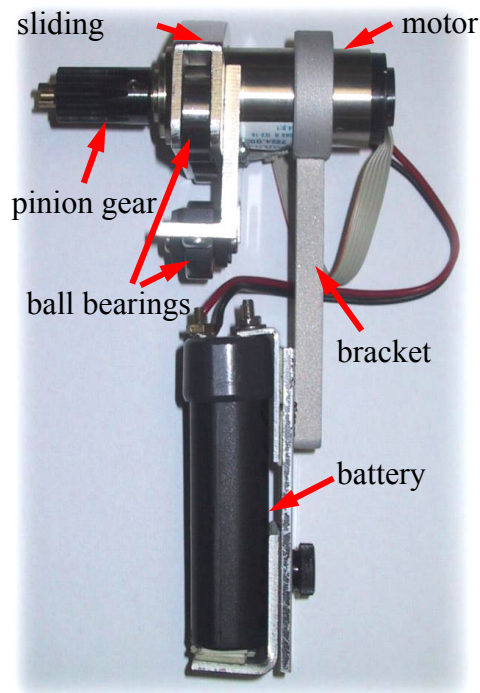


Figure 3.7: Side view of the sliding bed and the moving mass

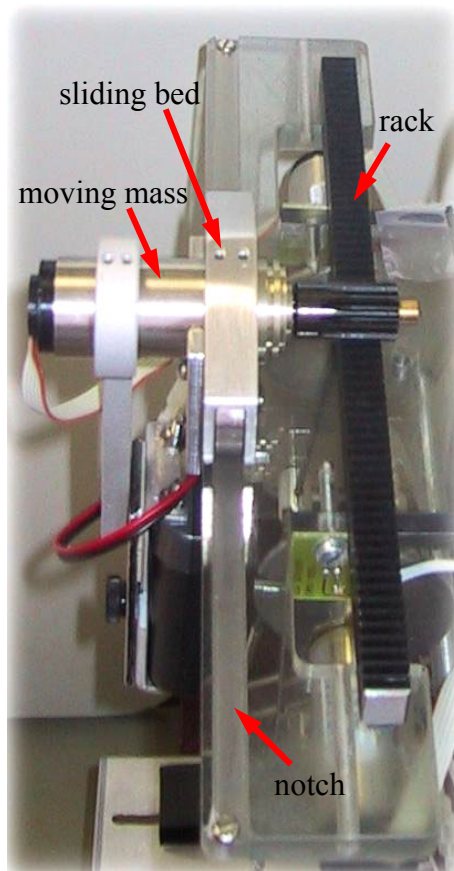


Figure 3.8: Actual design of the moving mass system

3.4. Foot design

The foot design was also changed a few times, starting with an attempt of a round foot (see Figure 3.9). At the MIT Leg Lab, many robots using curved or balled feet are able to balance on the spot. The only restriction to keep in mind is that the radius of the foot had to be less than the radius of the curvature of the walking path. However, during the test, this design causes a few problems. The control of the balance had to be very fast, to keep the system stable, because all position where unstable.



Figure 3.9: Curved foot prototype

An alternative foot was proposed and built in order to fix these problems. The ground plate of this passive¹ foot design shown in Figure 3.10 is made out of aluminium and the small block is made out of rubber. The support block is capable of propping up the robot when the foot touches the ground and the lower leg leans back on the block. With this arrangement, the robot had larger support area and a better control of the foot having positions where the required torque is not that high to keep it upright.



Figure 3.10: Actual passive foot

¹ The movement of this foot is not independent, because it has only a passive DOF

3.5. Prototypes and actual design

According to the previous chapters, the design of the whole robot changed many times. Therefore, the modelling and the controlling had to be changed as well, in order to adapt the model and the program to the new hardware. Each modification made improvements in the stability and the controlling properties. The Figures 3.11-3.14 show the evolution of this biped robot and the changes proposed and explained in the chapters before.

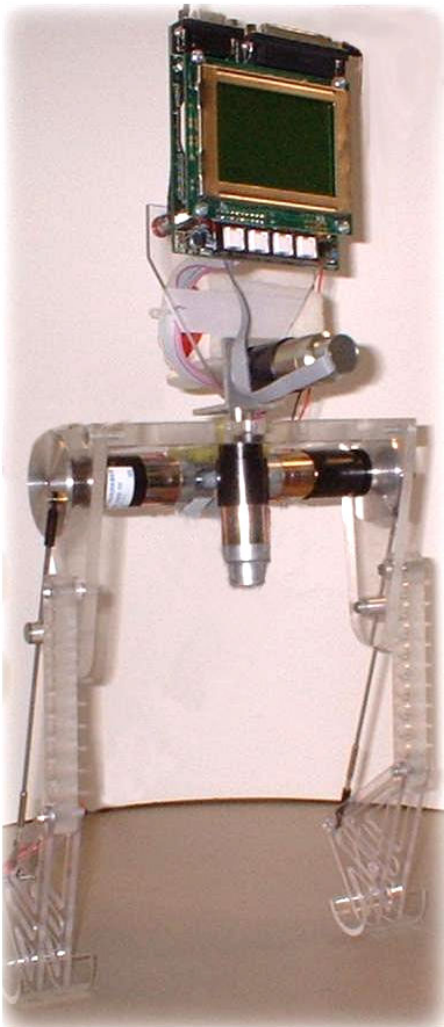


Figure 3.11: Prototype 1



Figure 3.12: Prototype 2

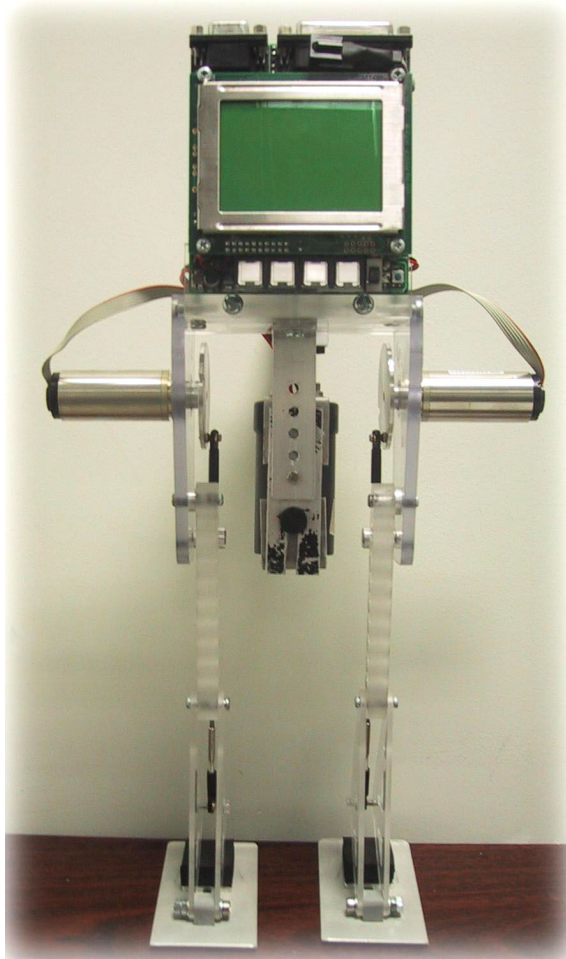


Figure 3.13: Prototype 3

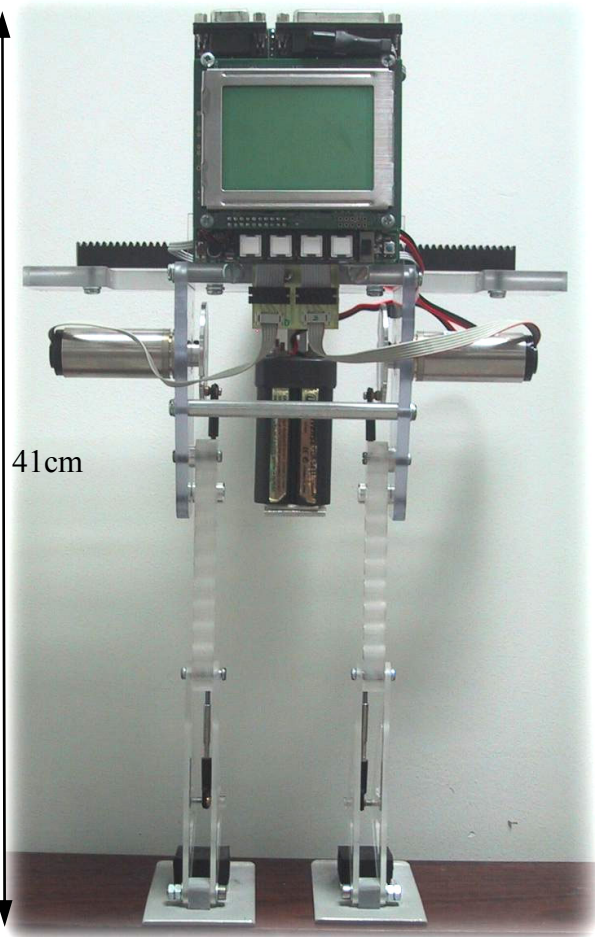


Figure 3.14: Prototype 4

4. Hardware and Software

4.1. The EyeBot

The EyeBot (see Figure 4.1) is a resourceful controller equipped with the 32-bit microprocessor 68332 from Motorola. The controller runs at a speed of up to 33 MHz and has a ROM of about 512kB. The different user programs can be stored in the RAM with a maximum capacity of 2048kB. For the user interface, there are four free programmable “softkeys” for user inputs and a Liquid Crystal Display (LCD) with 128x64 pixels for user output. The controller has a variety of ports including serial, parallel and several I/O pins. The processing capacity of the controller is also sufficient for most image processing tasks using a CCD colour camera as input, which can provide 32-bit RGB images at a resolution of 60*80 pixels. The ability to operate up to 12 servos or 4 motors with encoders makes it perfect for experiments with all kind of robots.



Figure 4.1: The EyeBot controller

4.2. Actuators

The actuators used in this robot are conventional DC motors (see Figure 4.2) provided with gearboxes and incremental magnetic shaft encoders. The gearbox with a transmission ratio of 54.6:1 provides enough torque to move the legs from every position, regardless which force is acting on the leg. Ignoring the properties of the gearbox, the maximum torque is:

$$T_{o-\max} = r \cdot T_{i-\max} \cdot \eta = 54.6 \cdot 0.005 Nm \cdot 0.73 \approx 0.1993 Nm \quad \text{equation 4.1}$$

The values are taken from the Faulhaber datasheets (Appendix C)

$T_{i-\max}$ = maximum torque input

$T_{o-\max}$ = maximum torque output

r = gearbox ratio

η = efficiency



Figure 4.2: Faulhaber DC Motor 2224R006S with inbuilt gearbox and encoders

The gearbox and the shaft encoders are integrated in the motorcase, so the motor size does not increase that much. These magnetic encoders are used for indication and control of both shaft velocity and direction of rotation as well as for positioning. The supply voltage for the encoder, the DC-Micromotor and the two-channel output signals are interfaced through a ribbon cable with connector. To connect the motors with the EyeBot, a small adapter PCB had to be built, because the pinout was incompatible.

The EyeBot has a Dual Motor drive onboard (see Appendix G), which can be controlled via software routines implemented in the BIOS of the robot (RoBIOS). To control two more motors, an extra PCB with another motor drive has to be attached to the controller, using the free Servo connectors as I/O and power supply. The output for the motors is made via a pulse width modulation (PWM) (see Figure 4.3). The voltage output is modulated using a PWM signal, with a clock speed of 8 kHz or even 16 kHz. Using this it is possible to control the speed of the motor very accurately, because the voltage is proportional to the speed of the motor.

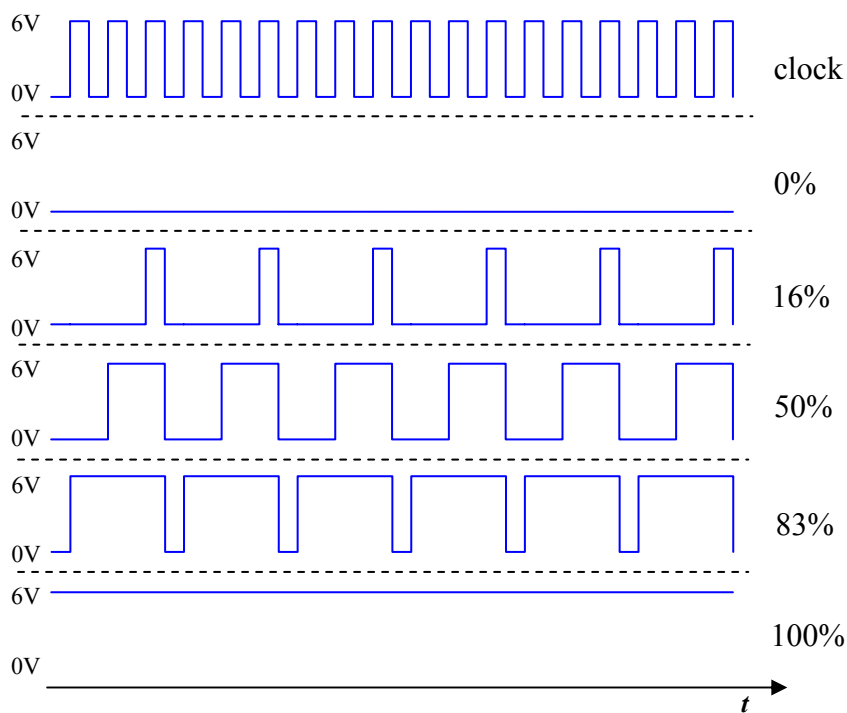


Figure 4.3: PWM signal at different values

Nevertheless, for an accurate closed loop position control of the motors, the actual current must be available. The current of a motor is proportional to the provided torque. Therefore, to maintain a certain position, the current needed must be known and controlled to provide a specific holding torque. The difficulties concerning this topic are explained in Chapter 5.

4.3. Sensors

To keep the robot as simple and cheap as possible, only a few sensors are used at this stage of development. The main information is provided by the encoders integrated in the motors (see Figure 4.2). They are two channel magnetic incremental sensors with 64 lines per revolution (see Appendix C for further details). Consequently, with the gearbox (gear transmission ratio = 54.6), attached to the motor, 3494.4 ticks mean one revolution of the shaft. This is a very precise feedback, because the accuracy is about $\frac{1^\circ}{9.7ticks}$. The counter can be read out at every point of the user program, using a special function provided by the RoBIOS.

However, because these sensors are incremental and not absolute, they must be reset at a defined position to have the absolute position of the feet. Once the feet are in a defined position, the counter of the encoders can be reset. Therefore, another sensor was needed to give feedback of the position for calibrating reasons. A small PCB was built, which was attached on the side plates of each leg. The components of the circuit were on the bottom of the PCB, whereas the sensor itself was fixed in a socket on the top of the board (see Figure 4.4). The OPB608B from Optek (see Appendix E for further details) is a reflective object sensor and consists of an infrared emitting diode and a NPN silicon phototransistor mounted aside on parallel axes. The phototransistor responds to radiation from the emitter only when a reflective object passes within its field of view. At this stage of development, the circuit for this consist of several resistors and a capacitor as low pass filter (see Appendix F for further details). Nevertheless, if necessary, the circuit could easily be upgraded, because it was developed with an additional Schmitt-Trigger circuit after the transistor output, to have a value-discrete output. These sensors were connected to the digital inputs of the EyeBot and could be read out by a RoBIOS function.

To have a defined mark, which could be identified by the sensor, a slot was cut into the crank and the sensor was placed in such a way, that it could detect the slot (see Figure 4.5). Now it was possible to program a calibrating routine, in which the motor was moved to this position and the counters of the encoders reset. After having reset the counters the position of the feet were absolute. Reading the values of the encoders, the position could be determined.

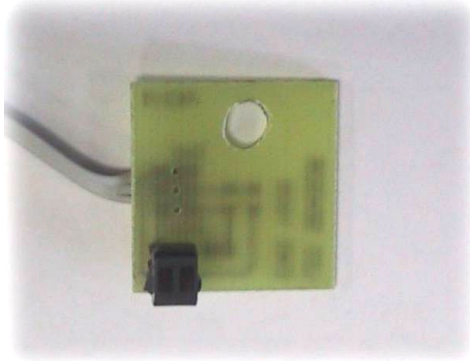


Figure 4.4: PCB with optical sensor

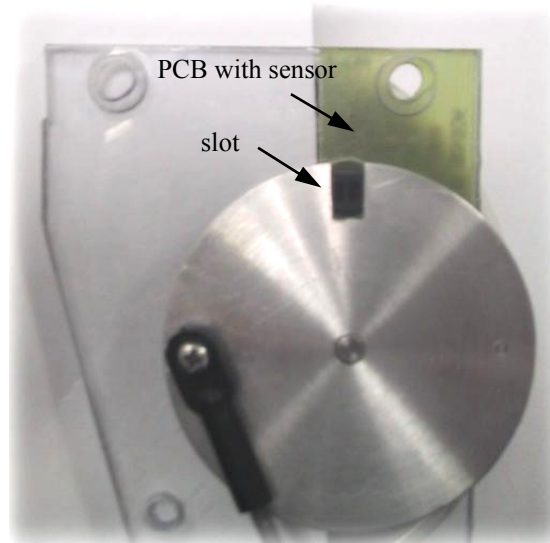


Figure 4.5: Side plate of one leg with slotted crank and optical sensor PCB

Furthermore, two inclinometers (see Figure 4.6) were proposed to measure the angle of the robot. The two sensors were mounted in such a way, that they could provide the changes of the angle of the frontal plane in respect to the vertical (pitch) and the changes of the angle of the sagittal plane in respect to the vertical (roll). The sensors N3 from Seika (see Appendix D for further details) have a maximum measuring angle of $\pm 30^\circ$, which is quite enough for this kind of application. This analog sensor has a voltage offset of 2.47V and sensitivity of $15 \frac{mV}{^\circ}$. Thus, the operating range of this output is about from 2.02V to 2.92V.

These sensors are connected to the analog inputs of the EyeBot. A single chip from Maxim MAX192 (see Appendix G for further details), integrated on the EyeBot, is used as analog digital converter (ADC) for these purposes. It has a built-in multiplexer, a T/H switch and a comparator and, can manage up to eight inputs. It has an accuracy of 10 Bits and has an internal 4.096V reference. Nevertheless, the time required for the T/H to acquire an input signal is a function of how quickly its input capacitance is charged. If the input signal's source impedance is high, the acquisition time lengthens and more time must be allowed between conversions. This results as a problem, because at higher speeds, the comparator was not able to deliver the right value while changing the channels. If the channel to read stays the same, the value was always correct, but if the

channel changes every time, as is the case if more than one sensor is attached to the inputs, the returned value from the ADC was not reliable. Thus, a software routine had to be implemented, that waits a certain time, to ensure that the correct data was returned from the ADC. The details of this software fix will be explained in Chapter 6.2.2.



Figure 4.6: Inclinometer

4.4. Software environment

The EyeBot can be programmed in assembler or in C/C++. The built-in BIOS, so-called RoBIOS, provides several C functions to access the different I/O and the connected hardware. The list of all the C functions integrated in the RoBIOS can be found at [6]. The user programs for the EyeBot are compiled on a PC, running Linux or Windows with a special for the Motorola 68332 modified version of the open source GNU C/C++ compiler. The program can be written with any kind of text editor and then be compiled and linked using the special commands of the compiler. Using a serial connection, the program can be downloaded to the RAM of the EyeBot. Once downloaded, it can be executed or stored in the ROM of the controller.

However, the compiler had not worked in the past with Windows XP. Nevertheless, the compiler should run under this operating system to be compatible to future software and upcoming operating systems.

The main problem was that Windows XP no longer supports real 16-Bit applications. While running such an application, the operating system simulates an old DOS environment with other memory distribution and the compiler had problems in allocating stack memory. The result was a stack allocation error. The original compiler was built in a real 16-Bit environment, compiling all libraries in this environment, and therefore the program wants the memory distribution of a real 16-Bit environment. Thus, running in a simulated environment, the compiler did not have the same conditions and was not able to compile.

Therefore, the source code of the GNU compiler, including all libraries, had to be compiled in a new environment similar to the one for Windows XP 16-Bit applications. Another open source program called Cygwin was able to provide a simulated UNIX environment under a Windows operated PC. The main part consists of a dynamic link library (DLL) that acts as a UNIX emulation layer providing substantial UNIX API functionality. Furthermore, different tools, ported from UNIX, provide a UNIX/Linux look and feel. After installing this program on a Windows XP machine, it was possible to compile the source code of the GNU C/C++ compiler under a simulated 16-Bit environment. Other difficulties were mastered in order to achieve a working compiler.

Furthermore, the modifications for the 68332 microcontroller had to be made to ensure that the compiled program would operate on the EyeBot. Therefore, all the libraries concerning the compiling and the linking also had to be adapted.

The new compiler and the new libraries were integrated with the other EyeBot libraries and header files. To ensure that the compiler would work on different machines, the compiler was tested on different platforms with the result that it works on almost every Windows operating system. The different batch files had to be adapted in order to be compatible to the DOS nomenclature and after having verified the whole bundle, an installation package for windows machines could be created. The package was created with the freeware tool Installer2Go and tested on different platforms.

To improve the user-friendliness, the compiler can now also be executed from any Win32 text editor, using specific batch files. After editing the C or C++ files in this Win32 application, the user does not need to leave the application and change to the command prompt to compile and link the files. The output of the DOS environment, in which the compiler is operating, can also be logged and displayed in the Win32 text editor. Even the download can be integrated into a Win32 text editor using a particular batch file.

The effort to rebuilt this compiler was rewarded with a user-friendly SDK that saves time while programming and makes it easier to work with, because it more similar to other SDK's. Furthermore, the ability to run on new operating systems makes it future-proof.

5. Modelling and Simulation

The structures and parameters of a system are normally described by mathematical models. Most of the methods used for the control system technology are based on these models. Once a model of a specific system is built, the system can be analysed, synthesized and optimized using just the model of it. If the model is accurate enough, the simulated controller can be implemented in the system, to create a precise close loop control of the complex system.

To get information about the structure of the system, there are two different types of procedures:

The theoretical analysis is based on the physical principles. These are essentially the laws of conservation of mass, energy and momentum and the laws of dynamic balance. The parameters itself depend on the physical properties of the system. The main precondition is a qualitative conceivability of different physical processes in the system. Knowing the system, the different laws can be applied and the system can be described as a qualitative model. Once the characteristic parameters of the system values are known, the equation can be formulated and the system-parameters can determined. In most case, the resulting model can be simplified and the number of differential equations can be reduced.

Using the experimental analysis, a model can be determined from measured input- and output-values. Having additional information, a convenient model of the system-structure can be defined. Then, the parameters can be extracted out of the known input and output values and the model. Usually, the model must be reviewed and adjusted where required.

The problem while trying to model a complex system like this robot is to manage the nonlinearity of the system. Most of the methods can only be applied, if the system is linear. Therefore, most systems must be considered as linear at the operating point and then the system can be modelled.

To construct a model of this robot the complexity of it was reduced. The problem was simplified to the legs. After having a working controller for the legs, the other parts of the system could be implemented. In order to achieve an exact model of the legs both methods explained before were applied and compared. After this, a proper controller could be simulated and its parameters improved. Finally, the controller could be programmed and integrated in the main program of the system.

5.1. Theoretical analysis of the legs

The main advantage of this method is that the system do not need to exist yet, it can be applied during the development period. The analysis provides information about the inner structure of the system and the real state equation. The disadvantage is that the model is normally very complex if an accurate model is desired. In addition, usually not all variables can be determined exactly and thus the model will be imprecise.

To start the analysis, the actuator of the whole system is modelled. The actuator for the legs is a DC motor and therefore the system must be analysed using the model of a motor. The moment equation describes the mechanical part of the system:

$$J_{tot} \cdot \frac{d\omega(t)}{dt} = M_A(t) = M_M(t) - M_F(t) - M_L(t) \quad \text{equation 5.1}$$

J_{tot} = total moment of inertia

ω = angular velocity

M_A = moment of acceleration

M_M = moment of the motor

M_F = moment of friction

M_L = load moment

Applying this equation to the actual problem keeping in mind that the system, consisting of the two dc motors, has two input and two output values, the following equations result:

$$\frac{d\omega_1(t)}{dt} = \frac{1}{J_1} [M_{M1}(t) - M_{T1}(t)] \quad \text{equation 5.2}$$

$$\frac{d\omega_2(t)}{dt} = \frac{1}{J_2} [M_{M2}(t) - M_{T2}(t)] \quad \text{equation 5.3}$$

ω_1 = angular velocity of the left leg

ω_2 = angular velocity of the right leg

J_1 = total moment of inertia of the left leg

J_2 = total moment of inertia of the right leg

M_{T1} = total moment of friction of the left leg $\rightarrow (M_{T1} = M_{F1} + M_{L1})$

M_{T2} = total moment of friction of the right leg $\rightarrow (M_{T2} = M_{F2} + M_{L2})$

with $M_M(t) = K_M \cdot u(t)$ equation 5.4

and $M_T(t) = r_C \cdot \omega(t)$ equation 5.5

K_M = constant of the motor

u = armature voltage

r_C = attenuation coefficient

Assuming that two equal motors have the same motor constant the equation 5.2 and 5.3 can be expressed as follows:

$$\frac{d\omega_1(t)}{dt} = \frac{1}{J_1} [K_M \cdot u_1(t) - r_{C1} \cdot \omega_1(t)] \quad \text{equation 5.6}$$

$$\frac{d\omega_2(t)}{dt} = \frac{1}{J_2} [K_M \cdot u_2(t) - r_{C2} \cdot \omega_2(t)] \quad \text{equation 5.7}$$

u_1 = armature voltage of the left motor

u_2 = armature voltage of the right motor

r_{C1} = attenuation coefficient of the left motor

r_{C2} = attenuation coefficient of the right motor

According to equation 5.6 and 5.7 the block diagram looks as follows:

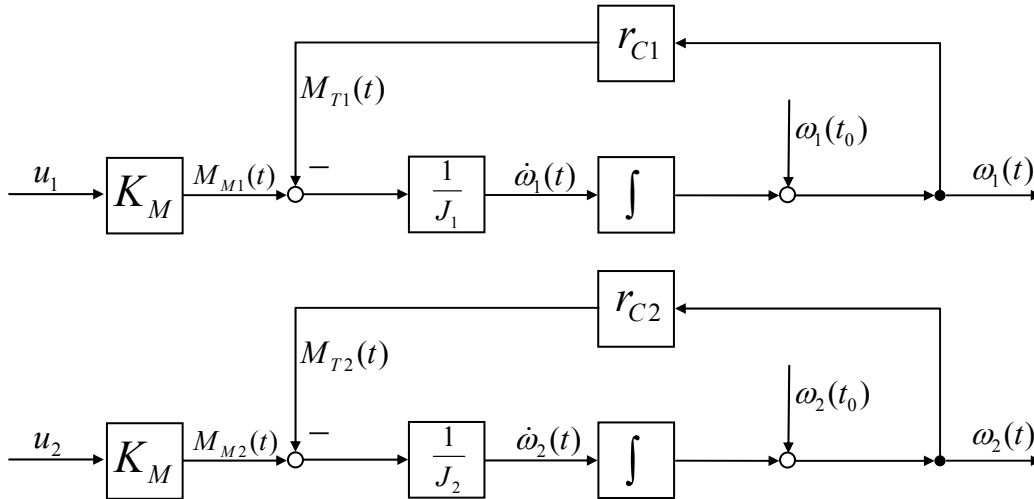


Figure 5.1: Block diagram of the two legs

At this point, a theoretical model for both legs was built using the differential equation of the motor. The next step would be to determine the different constants of the system. However, the difficulties were that some constants could not be measured, because the adequate measuring instruments were not available or a calculation of it would be too time-consuming. Therefore, some constants were approximated and, using the experimental analysis explained in the next chapter, optimized. Nevertheless, the friction constant r_c is not constant at all. The force acting on one leg changes depending on the position of the foot. This fact will be explained and considered further on, when the controller is designed in the simulation. In addition, the momentum constant K_M is non-linear and depends on the speed of the motor. Thus, a desired operating point must be defined and at this point, a linear approximation can be done. The desired motor speed will be defined in the next chapter having regard to the advantage and disadvantage of the different speeds.

Another point to be considered is the input voltage. The model illustrated in Figure 5.1 only works ideal, if the input voltage is a continuous value. However, the motor drive of the EyeBot provides a PWM signal at a high frequency as input voltage for the motor, which also can have a nonlinear influence on the system. Therefore, this has to be integrated in the simulation in order to obtain a good model of the system.

5.2. Experimental analysis of the model

The advantage of this kind of analysis is that there is no need to understand the physical interrelationship of the system to obtain a model of the system. Furthermore, the resulting mathematical model and its transfer function are in most cases of a low order. Nevertheless, the system must already exist to use this method and it provides no information about the inner state of it.

To find an appropriate model of the system using the experimental analysis, the system must be without kinetic energy. At time $t=0$ a signal u_y will be applied to the input [9]. This is equivalent to applying the step function to the input.

$$u_y(t) = u_{y0} \cdot E(t) \quad \text{equation 5.8}$$

$$u_y(s) = u_{y0} \cdot \frac{1}{s} \quad \text{equation 5.9}$$

To identify the structure of the system, the progress of the output $x(t)$ is measured and this step response is compared with other step responses from known transfer elements. The transfer function of the system would look as follows:

$$G_S(s) = \frac{x(s)}{u_y(s)} \quad \text{equation 5.10}$$

For a linear time invariant (LTI) system the input is proportional to the output. Consequently, only one measurement is necessary to determine the function. However, if the system is not linear, multiple measurements must be taken in order to obtain enough information to find a linear approximation of the system [10].

Therefore several measurements with one motor at different speed were taken to achieve an adequate step response. Only one motor is researched in this chapter, because the model of the second motor would be equal to the one obtained.

A small program was created that generated the step function for the motor and logged all the data of the output. That means that the motor was abruptly turned on and the leg starts moving for about 2sec. Meanwhile the position of the motors was determined with the encoders and the data obtained from the encoder was stored in an array. The sampling frequency of the system was about 100Hz because the interrupt-function, in which the program runs, was called every 0.01 seconds. This had to be considered later on because the controller is slightly different in a sampling system. The logged data was then transfer with serial connection to the computer and stored in a file. Finally, the file was imported to Excel and the needed calculations were made. The measurements were taken at a speed of 20%, 30%, 50%, 70%, and 80% of the PWM. Furthermore, different weights were applied to the leg in order to change the force acting on the system. All the results were analysed and for each step response, a transfer function was determined applying the 63%- and the tangent-method. This approximation is normally used for lag dead time systems and systems with motors as actuators can often be described with such a model. The constructed model was accurate enough, as the comparison with the theoretical model will show.

The determination of the transfer function will be shown at the step response at a speed of 80% of the PWM. This point was chosen as the operation point for following reasons. The system will be fast enough for a dynamic walk, the torque provided by the motor was high enough to keep the leg in motion albeit the counterforce acting on it, and there was still a speed reserve if the leg had to move faster in order keep a stable dynamic walk.

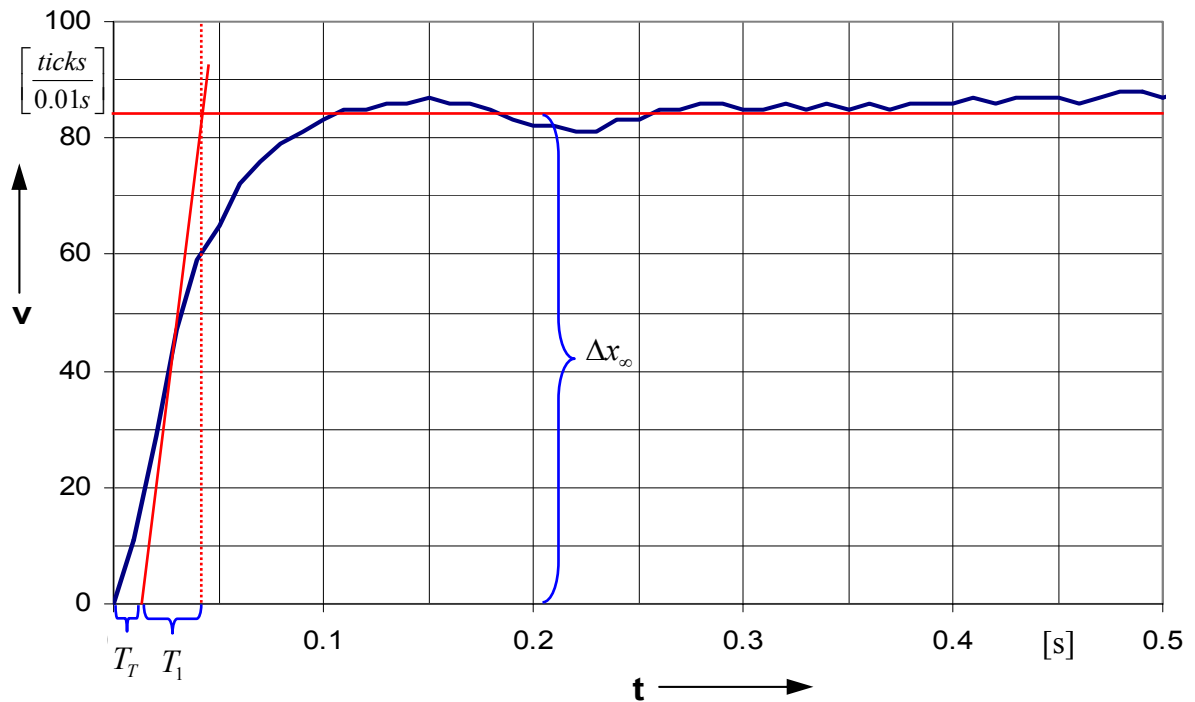


Figure 5.2: Step response at 80% of the PWM with tangent method parameters

The transfer function of a lag dead time system is shown in equation 5.11:

$$G(s) = \frac{k}{1 + T_I \cdot s} \cdot e^{-T_T s} \quad \text{equation 5.11}$$

k = proportional gain

T_I = integral time

T_T = dead time

To obtain the parameters of the transfer function, a different calculation had to be made:

$$k = \frac{\Delta x_\infty}{\Delta y} \quad \text{equation 5.12}$$

Δx_∞ = delta output after infinite time

Δy = delta input

The input voltage was 80% of the PWM and the output can be read out of Figure 5.2 with a value of $85 \frac{\text{ticks}}{0.01 \text{ s}}$. The parameter T_1 , also read out from Figure 5.2 is about 0.039s. The dead time T_T is about 0.014s. For further calculations, the unit will be omitted to provide a better overview.

$$k = \frac{85}{80} = 1.0625 \quad \text{equation 5.13}$$

With these values, the transfer function can be written as follows:

$$G(s) = \frac{1.0625}{1 + 0,02 \cdot s} \cdot e^{-0,014 \cdot s} \quad \text{equation 5.14}$$

This is the system transfer function for one leg. The system can now be modelled and simulated in a software environment. A proper controller can be chosen, and its stability tested and improved.

Another possibility that was chosen to determine the transfer function of the system was the use of the software WinFACT. The module Ida from WinFACT is able to approximate a given step response and calculate its transfer function. Figure 5.3 shows the step function and the response of the system (both displayed dashed). The step response of the system shows a periodic disturbance added to the systems output which has his origin in the different forces acting on the leg while moving. The solid line in green was the approximation generated by WinFACT. The calculated transfer-function of the approximation is shown in equation 5.15. The result is quite similar to the one found using the tangent method. This consolidates the point of view that the dead time of this system is quite short.

$$G(s) = \frac{1.07}{1 + 0,02 \cdot s} \cdot e^{-0,01 \cdot s} \quad \text{equation 5.15}$$

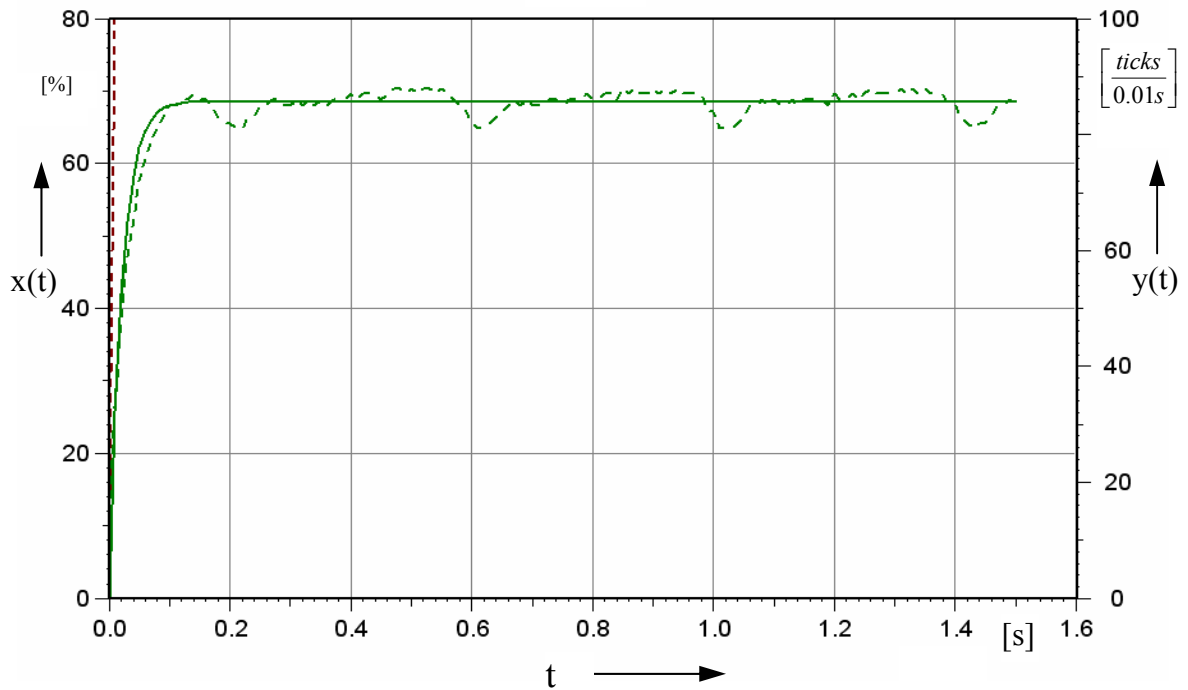


Figure 5.3: Step-function, step-response and approximation of it using WinFACT

Having the information of both theoretical and experimental analysis, the whole system can be simulated with a generated level of accuracy. The unknown parameters from the theoretical analysis can now be found out, approximating the step-response of theoretical model to the one of the experimental.

The different controller designs for a close loop control can now be tested and after finding a suitable design for this special system, its parameters can be optimized.

5.3. Choice of a controller for close loop control

Different types of control methods are available. Depending on the application, some are more appropriate than others are. In this chapter, different controller types advantages and disadvantages are described in order to explain the reason for the used type for this particular system.

Like usual for systems with several inputs and outputs, a controller concept at the state-space is chosen. This powerful tool to design a controller is normally chosen for such systems because an analysis with the classic methods would result in a high order model and a complex transfer function. However, the peculiarity of this special system is that it is a compound of two similar single systems, one for each leg. Only a few parameters differ. That means that it can be considered as two independent systems. Therefore, a controller design using classic methods was chosen, simulating two independent controllers, one for each leg. Furthermore, a simpler controller would also keep the computation time low.

Having a sampling system, the use of a dead-beat control is suggestive. The controller can be designed in such a way that at a given time the output has the exact value of the desired value. The controller transfer function is chosen in a way such, that the poles of the system will be compensated at a finite time. The minimal time that can be chosen, depends on the order of the system and the sampling period. Nevertheless, the dead-beat controller has the problem that it will deliver high values of the correcting variable. If the actuator is not able to provide such high values, the finite time of the controller has to be extended. Because the operating point was chosen at about 80% of the maximal output, the finite time of this controller has to be long, to avoid an output above 100%. However, such a long control time was not desired and therefore the dead-beat control was discarded.

Finally, the decision to take a PID-control as the control algorithm was taken. This controller is comparatively easy to implement and the computation time would be short.

Another point to keep in mind is the values to be controlled. The difficulty of this is, that the speed and the position of the legs needs to be controlled. The only feedbacks provided by the actuators are the counters of the encoders. From this information, the position and the actual speed (calculating the difference between two positions) can be extracted. However, the actual current is not available and that complicate trying to keep an exact position. The actual flowing current is proportional to the torque provided by the motor. Therefore, for accurate positioning only a specific torque and no movement of the motor-shaft is required.

It was discovered, that for the dynamic walking the best way to control the legs is to keep a 180° phase-shift between the legs. That means, that the movement of one leg, independent from the actual speed and the force acting on it, has to be at any given time a half turn behind the other leg. The biped would then have at any given time a defined feet position, no matter if it is stopped or in motion. Consequently, the control for the counterweight would be easier, because it has no regard to the actual speed of the legs. The difficulty of this kind of leg control is that the position has to be controlled using the speed as the manipulated variable. The controller had to be able to maintain the speed and phase-shift position of the legs.

5.4. Simulation with WinFACT

The simulation of the leg model was done with the module BORIS from WinFACT. The elements can be easily added, tested and the step response evaluated. Furthermore, the built controller can be optimized using the different optimizations parameters.

5.4.1. Implementation of the model

First, the model of the theoretical design (see chapter 5.1) is built using the standard element of BORIS (see Figure 5.4). The desired value is the speed, and it can be altered using the PWM signal. Thus, the input is the voltage of the motor and the output is the angular velocity. The constant of the system, specially the friction constant is estimated knowing the step response from the experimental model. By comparing and approximating the step response of the model shown in Figure 5.4 with the simulated step response, the response of the experimental model and all the unknown values can be determined.

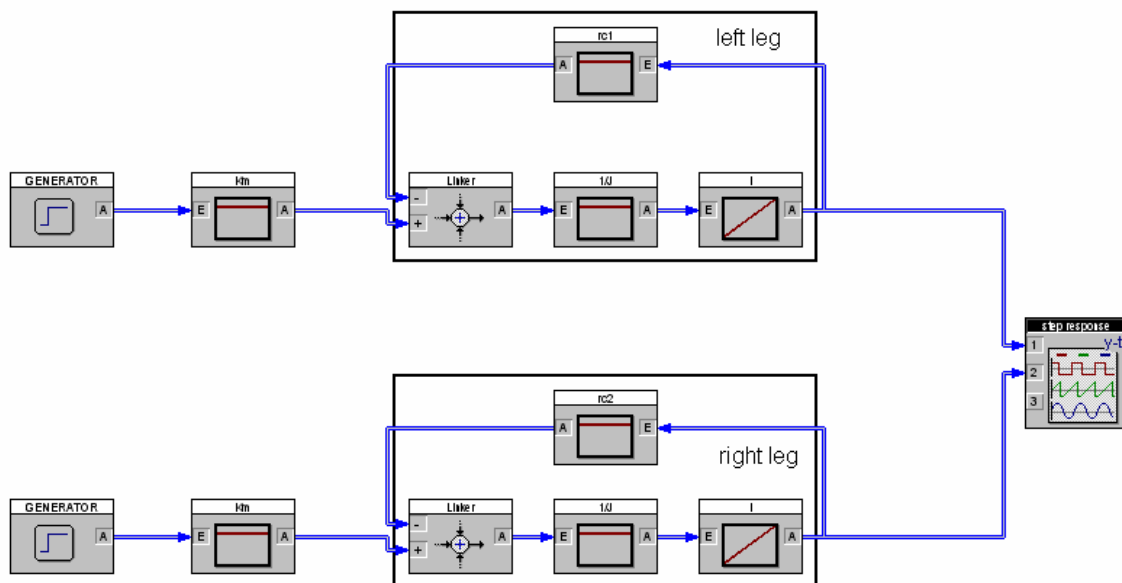


Figure 5.4: Block diagram of the open loop system

5.4.2. Implementation of a controller

After having the exact parameters of the whole system, the controller can be implemented. Figure 5.5 shows the block diagram of the system with a close loop control for each leg. Two independent PID controllers are integrated, but because they are controlling identical systems, they have the same parameters. To generate the interaction between both legs systems, the difference of the positions (taking into account the 180° phase shift) from each leg was built. After multiplying this value with a constant, it was given back to the input of each system. Hence, the needed feedback from each system was applied to the input of both. In addition, a disturbance affecting the output of each subsystem was added (the disturbance of the real system can be seen in Figure 5.3). This disturbance reproduces the counterforce acting on each leg. Because of the foot movement is nearly on an elliptic trajectory, the force acting on it can be estimated as a sinusoidal oscillation. Therefore, the disturbance was provided by a waveform generator, which produces a sinusoidal signal. To realize the PWM voltage, the input was provided by a waveform generator.

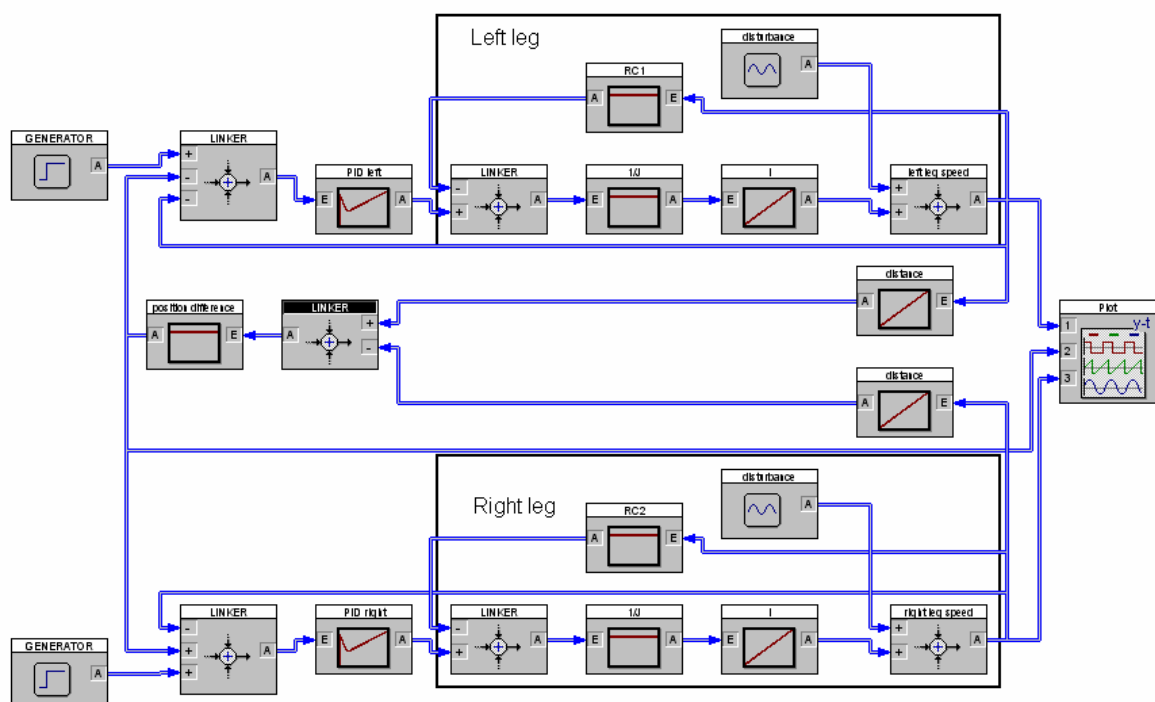


Figure 5.5: Block diagram of the close loop control of the system

5.4.3. Optimisation of parameters

The parameters of the PID-controller were optimized using different methods. The classic methods used for this kind of controller are for example the rules from Chien, Reswick, Hrones, or from Ziegler and Nichols. Furthermore, there are other methods like the amplitude optimum or the symmetrical optimum. They all have tables on which the parameters of the controller can be calculated. They used different techniques and diverse methodology at the frequency domain, to get to those rules.

However, depending on the system, not all the methods can be applied. For this system, the rules from Chien, Reswick, Hrones were applied. Although, with this scheme, the controller may tend to overshoot, it was utilized because of its ability to provide a robust and fast controller for first order systems, even if the system leaves the linearised operation point. The main thing is that the controller had to be fast, in order to keep the legs at the desired position even if the speed would temporally be very high. Therefore, an overshoot was tolerated.

The transfer function of a PID-controller is shown in equation 5.16:

$$G(s) = k_p \cdot \left(1 + \frac{1}{T_i \cdot s} + T_d \cdot s \right) \quad \text{equation 5.16}$$

k_p = proportional gain

T_i = integral time

T_d = differential time

Following parameters for the controller were determined using the rules (the unit are omitted for a better overview):

$$k_p = \frac{1.2 \cdot T_1}{k \cdot T_T} = \frac{1.2 \cdot 0.025}{1.0625 \cdot 0.014} \approx 2.02 \quad \text{equation 5.17}$$

$$T_i = 2 \cdot T_T = 2 \cdot 0.014 = 0.028 \quad \text{equation 5.18}$$

$$T_d = 0.42 \cdot T_T = 0.42 \cdot 0.014 \approx 0.0059 \quad \text{equation 5.19}$$

After having applied this method, the parameters were fed to the two controllers of the simulated system and the systems response was analysed (see Figure 5.6). The red curve is the left leg, the blue is the right leg and the green is the error position. After 0.1s, the sinusoidal disturbance for both legs was turned on, with a phase shift of 180° for one leg, to simulate the different leg movement. The diagram shows that the controller can rapidly achieve the desired speed of 80 and is able to keep this value stable even if a disturbance is present.

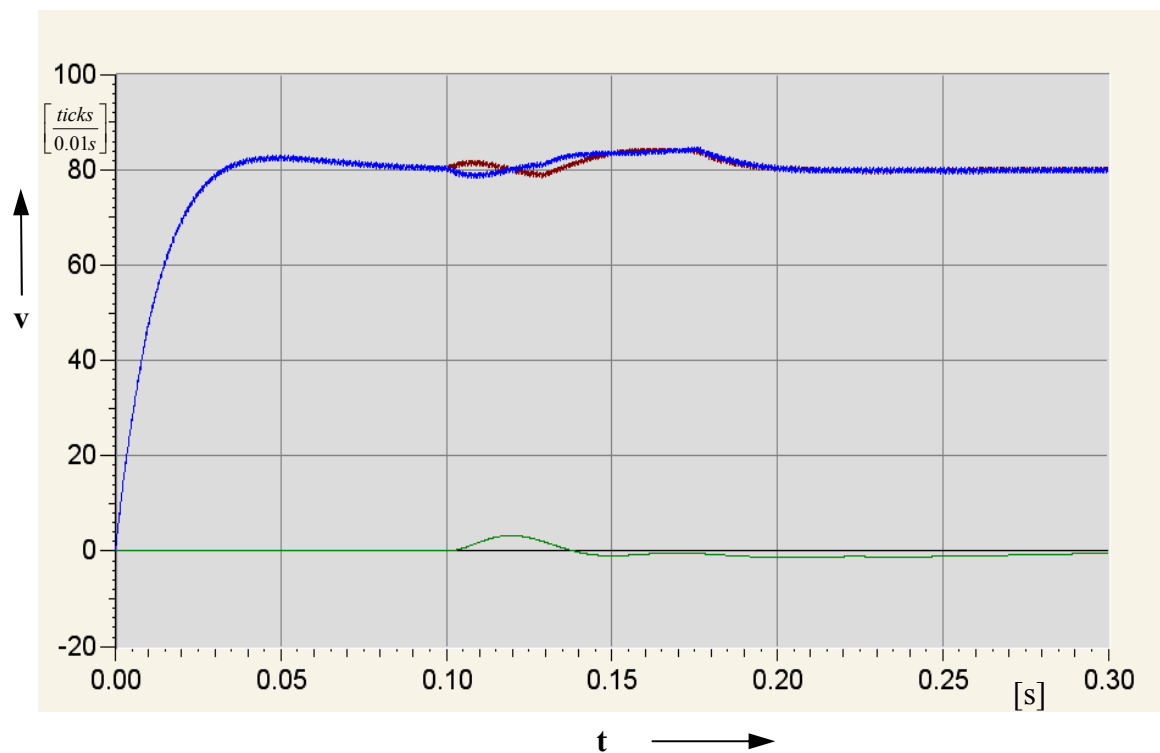


Figure 5.6: Step response using a close loop control with activated disturbance

The controller factors could be improved by using the optimisation function of WinFACT. After entering the optimisation parameters, the program was able to improve the controller parameters. The program uses a genetic algorithm and after a desired number of generations, it finishes the improvement process. The result is mostly an enhancement of the controller parameters as the graph in Figure 5.7 shows.

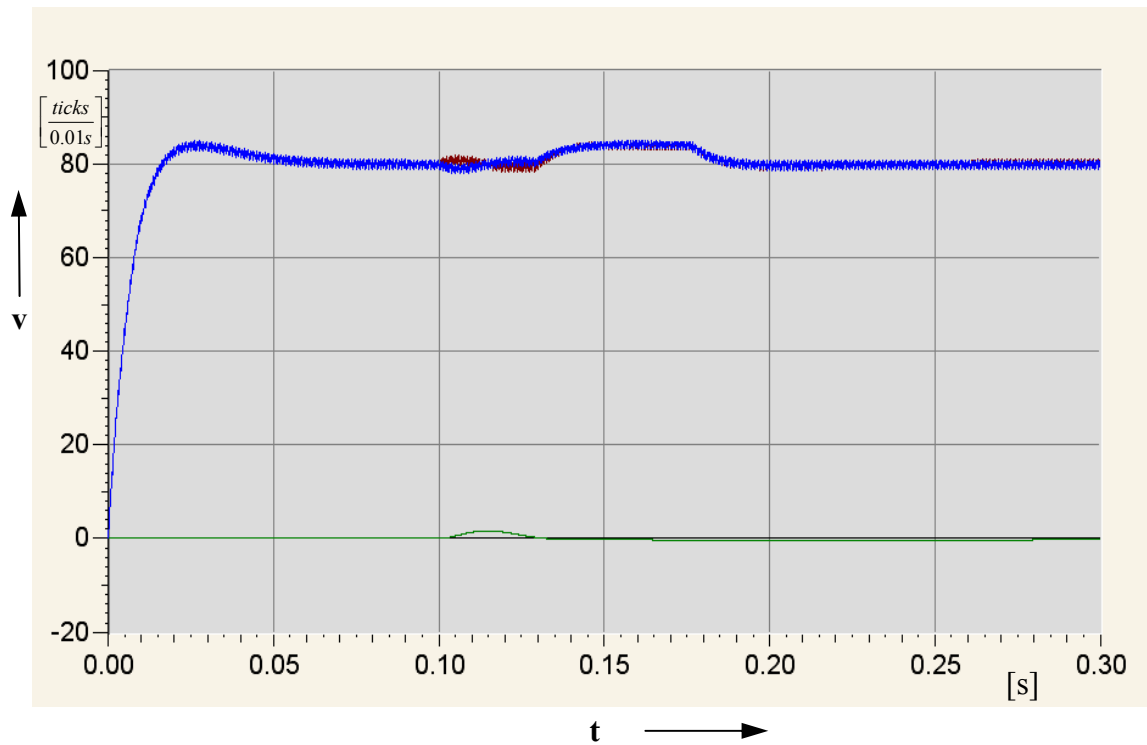


Figure 5.7: Step response using the optimized close loop control with activated disturbance

The system now reacts even faster to changes of the desired value. Now the rise time is nearly half of the time as it was before optimizing. Furthermore, the disturbance compensation still works accurately enough.

The optimized parameters for the PID controller are finally:

$$k_p = 3.95 \qquad T_i = 0.015 \qquad T_d = 0.005$$

5.5. Realization of the close loop control with C++

After having built a precise controller for this model, it could be converted into a software control algorithm. Therefore, the continuous system has to be transformed into a sampling system.

Consequently, the controller itself has to be transformed. The common equation for an analog PID-controller is shown in equation 5.20

$$y(t) = k_p \cdot \left[x_d(t) + \frac{1}{T_i} \int x_d(t) dt + T_d \cdot \frac{dx_d(t)}{dt} \right] \quad \text{equation 5.20}$$

x_d = error value

Converting the continuous values to discrete values and the integrals and differentials into differences, the algorithm looks like follows:

$$y_k = y_{k-1} + k_p \cdot \left[\left(1 + \frac{T_d}{T} \right) x_{d,k} - \left(1 - \frac{T}{T_i} + 2 \frac{T_d}{T} \right) \cdot x_{d,k-1} + \frac{T_d}{T} \cdot x_{d,k-2} \right] \quad \text{equation 5.21}$$

y_k = actual output
 y_{k-1} = output one step previously
 $x_{d,k}$ = actual error value
 $x_{d,k-1}$ = error value one step previously
 $x_{d,k-2}$ = error value two steps previously
 T = sampling time

During desired-value steps, the differential component of the algorithm can achieve high values that can lead to oscillations of the whole system. Thus, an algorithm, which only acts on the desired value and not on the error value can minimise this. That means that for the D- component of the algorithm, x_d is substituted by the desired value x . Equation 5.22 shows the modifications.

$$y_k = y_{k-1} + k_p \cdot \left[x_{d,k} - x_{d,k-1} + \frac{T}{T_i} \cdot x_{d,k-2} - \frac{T_d}{T} \cdot \left(x_k - 2 \cdot x_{k-1} \frac{T_d}{T} + x_{k-2} \right) \right] \quad \text{equation 5.22}$$

In the program all the needed variables were added and initialized, and a function `feetcontrol` was created. An extract of the function is shown below, including the data acquisition, the calculation of the needed values and the calculation of the outputs for the motors. This control function was called in an interrupt routine every 10ms (for further details see Chapter 8). A few values had to be adapted or modified, in order to improve the controller properties when the system is far away from the operation point, and therefore nonlinear. At this stage the algorithm comprises only the P- and I- components of the controller to avoid high overshooting provided by the D-component.

Nevertheless, the result of this algorithm was that the legs of the robot keep moving at a desired speed and at the desired phase shift, regardless the forces acting on the leg. If the counterforce was for any reason too high for the motor to keep the speed and the leg was stopped, then the other leg stopped immediately, because in order to keep the balance of the robot, the position takes priority over the speed of the legs.

```
// get inputs
p_l = QUADRead(qH[0]);
p_r = QUADRead(qH[1]);

// get speed
s_l = ( p_l_old - p_l ) * 5 / SCALE;
s_r = ( p_r_old - p_r ) * 5 / SCALE;

// error position
e_p_l = (p_l_old - p_r_old);
e_p_r = (p_r_old - p_l_old);

// error speed and error position
e_s_l = w_s_l - s_l - (e_p_r * e_pp); // clickspersec
e_s_r = w_s_r - s_r - (e_p_l * e_pp); // clickspersec

// calculate outputs
o_l = (o_l_old + Kp * (( e_s_l - e_s_l_old) + ( c_1 * (e_s_l_old))));
o_r = (o_r_old + Kp * (( e_s_r - e_s_r_old) + ( c_1 * (e_s_r_old))));

// outputs to motor
MOTORDrive(mH[0], - (Round ((o_l < 100) ? ( (o_l > 0) ? o_l : 1) : 100)));
MOTORDrive(mH[1], - (Round ((o_r < 100) ? ( (o_r > 0) ? o_r : 1) : 100)));

// variables to old
p_l_old = p_l;
p_r_old = p_r;

o_l_old = Limit ((int) (o_l), 100, 0);
o_r_old = Limit ((int) (o_r), 100, 0);

e_s_l_old = e_s_l;
e_s_r_old = e_s_r;
```

Code 5.1: Extract from the PI-control algorithm

6. Static balancing control

6.1. Centre of mass

After a suitable control of the leg movement was programmed, the program for the total control system could be designed. First, a static balancing control was proposed. Only if the system was able to maintain a stable position, regardless which forces were acting on it, dynamic control was suggestive. A very important point to keep the balance of a biped robot is to know where the actual centre of mass is. The static stability can then be measured from the system's kinematic configuration. The static stability margin (SM) is the shortest distance between the projected centre of mass of the system and the boundaries of the support polygon formed by the convex hull of the supporting feet [11]. The principal defects of static stability measures, though, are that they do not take into account velocity, inertial effects, and the influence of the swinging legs and their future ground contact.

To determine the exact position of the COM, measurements with different positions of the counterweights were made. The results of these calculations were also used to change the design of the robot and to determine the convenient weight of the moving mass and the adequate position of it.

To calculate the COM of any given object the density distribution and the dimension must be known.

$$x_s = \frac{\int x \cdot m(x) dx}{\int m(x) dx} \quad y_s = \frac{\int y \cdot m(y) dy}{\int m(y) dy} \quad z_s = \frac{\int z \cdot m(z) dz}{\int m(z) dz} \quad \text{equation 6.1}$$

Most of the objects have homogeneous density or at least it can be approximated so, which simplifies the calculation (see Equation 6.2)

$$x_s = \frac{\sum x \cdot \Delta V}{V} \quad y_s = \frac{\sum y \cdot \Delta V}{V} \quad z_s = \frac{\sum z \cdot \Delta V}{V} \quad \text{equation 6.2}$$

For easier calculations, the equations were adapted to areas (see equation 6.3), in order to calculate the positions of the COM at the frontal plane and the sagittal plane separately.

$$x_s = \frac{\sum x \cdot \Delta A}{A} \quad y_s = \frac{\sum y \cdot \Delta A}{A} \quad z_s = \frac{\sum z \cdot \Delta A}{A} \quad \text{equation 6.3}$$

Part	Material	Quantity	Single Mass [g]	Total Mass [g]
Motor		3	130	390
Eyebot		1	238	238
Foot	Aluminium	2	32	64
Battery + bracket		1	160	160
Cograil	Steel	1	138	138
Cogwheel	Steel	1	22	22
Motor bracket	Aluminium	1	72	72
Optical sensor		2	6	12
Inclinometer		1	24	24
Side plate	Perspex	2	38	76
Hip plate	Perspex	1	92	92
Leg Mechanism	Perspex + Aluminium	2	58	116
Controller bracket	Perspex	1	20	20
Total Mass				1424

Table 6.1: Measured weights for the prototype 4

The individual parts of the robot were weighed (see Table 6.1) and the COM of each part was calculated. This allows, the entire COM to be calculated. The calculation were made using Excel. Drawings of the frontal and sagittal plane of the prototype 4 were made at a scale of 1:2, and added to the graphs, in order to have a better overview of the location of each point. By moving different parts of the system or changing their weight, it was possible to view the alterations of the position of the COM instantly, without the need to test the real system. This saved not only time but also mechanical work, and the prototype could be directly adapted to the calculated and optimised system. It was also possible to see the effect of the moving mass system. The calculations were also made for the moving mass system moved to the left and right limiter to figure out the displacement of the COM.

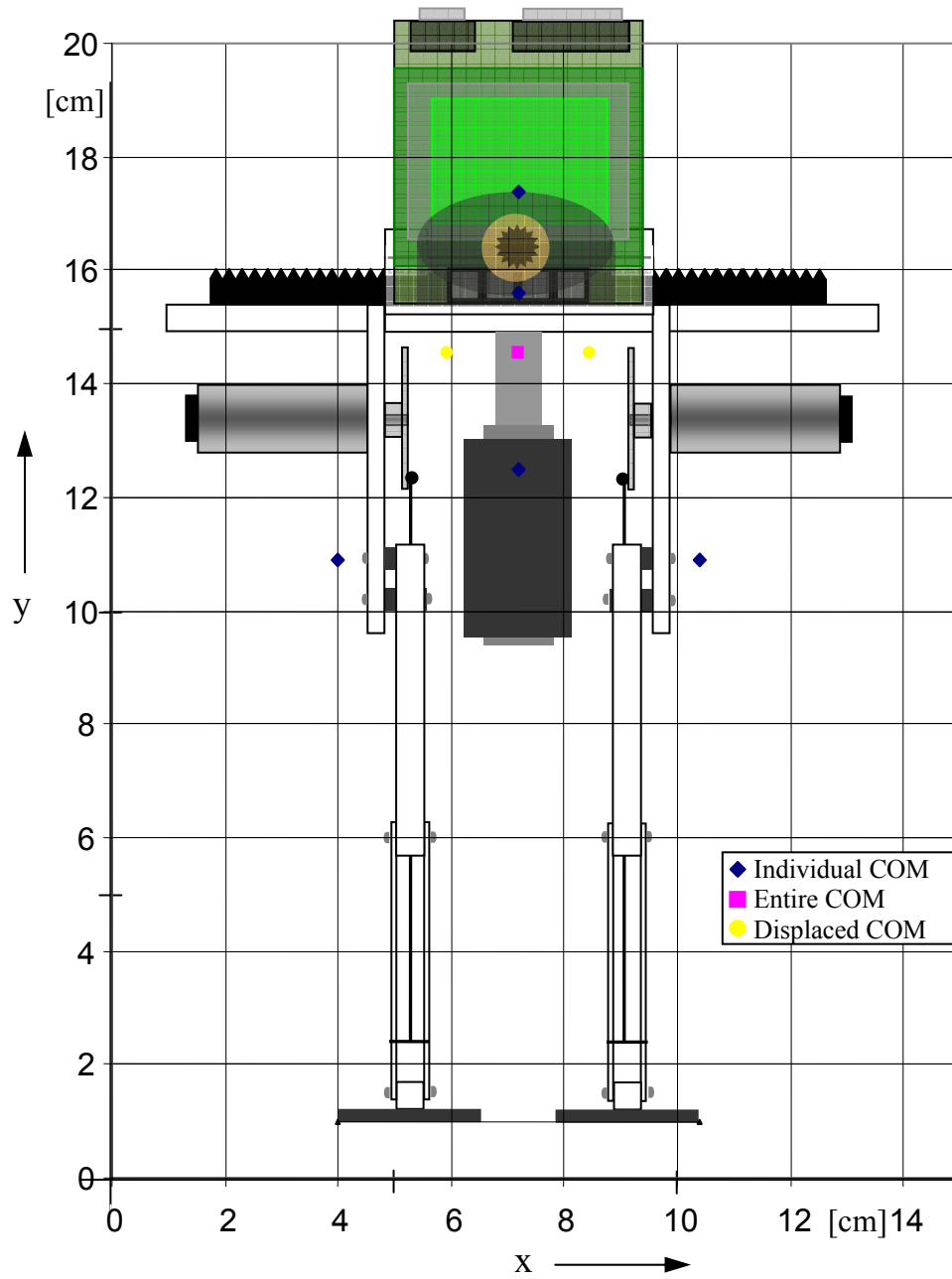


Figure 6.1: Graph of the frontal mass distribution and COM position (scale 1:2)

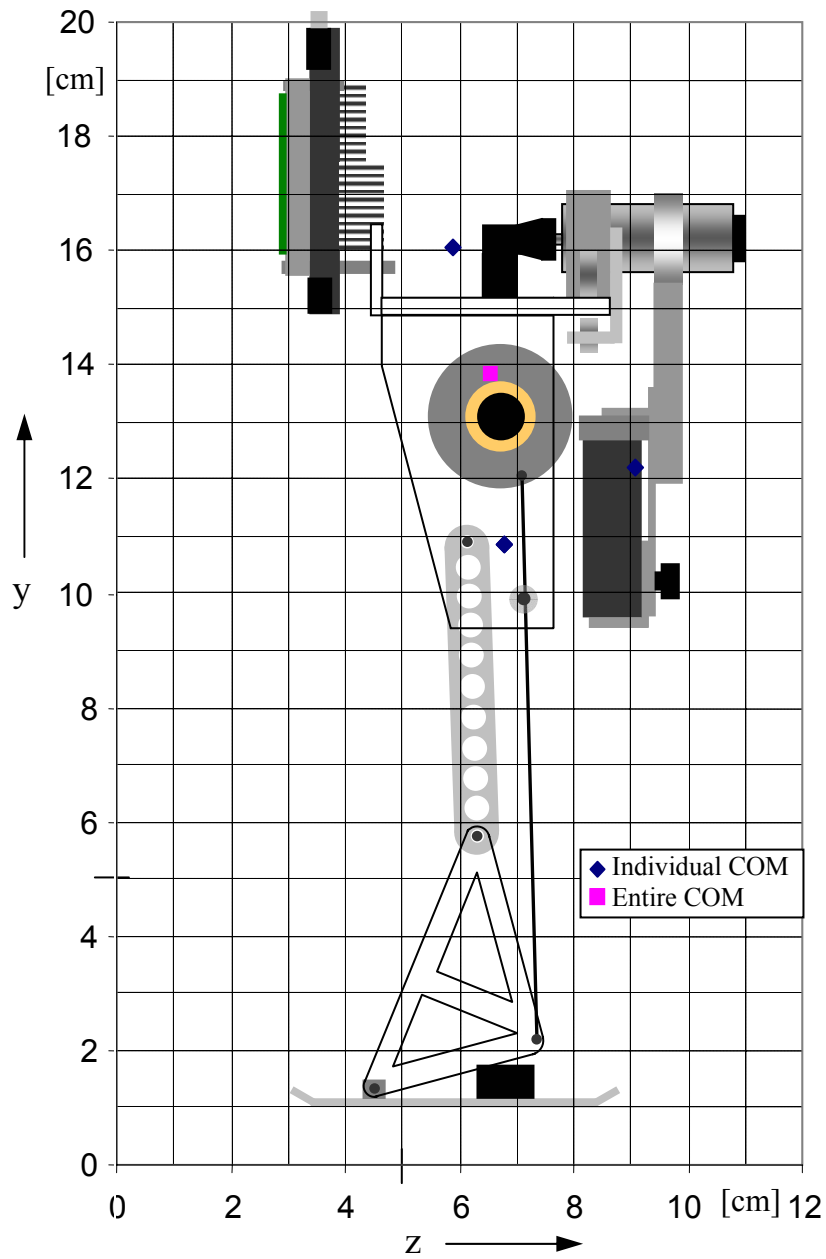


Figure 6.2: Graph of the sagittal mass distribution and COM position (scale 1:2)

As Figure 6.1 shows, the COM can be moved about 0.75 cm in each direction using the moving mass. The COM can be displaced in such a manner, that the NPCM of it is in between the support area of a single foot. That means that the whole robot could be balanced on a single foot if this is needed.

The Figure 6.2 shows that the mass distribution was designed in such a way, that the NPCM is nearly always in between the support area of the foot. If necessary, the position of the battery bracket can be changed forwards or backwards in order to alter the COM in the Z-axis.

After knowing the position of the COM, the control for the system could be designed to keep this point in between the support area. Thus, several sensor feedbacks had to be considered, processed and interpreted the right way for an apt close loop control.

Once the COM of the system is known, the relation between this point and the angle of roll could be determined. As the angle of roll stays in direct relation with the position of the COM at the sagittal plane, this value could be easily used to control the moving mass, because a real time computation of the actual COM would take a lot of computation time and this angle could easily be provided by an inclinometer. The advantage of this method is, that enough computation time is still available.

6.2. Sensor feedback

The various sensors used with this robot are mentioned before in chapter 4.3. In this section, a short overview of the signal input and signal processing is provided.

6.2.1. Foot feedback

As mentioned in Chapter 4.3 two different sensor feedbacks were available from the legs. The main information is provided by the built-in encoders of the motors. Each motor uses two of the 16 available TPU channels of the 68332 microcontroller, to allow for bidirectional measurement. The number of ticks is stored in an array, which can be retrieved via a RoBIOS function. The accuracy of these counters is very high, and thus makes them suitable to determine not only the actual position but also the actual speed of each leg mechanism, by means of finite difference approximation.

On each leg an optical sensors was attached similar to those described in Chapter 4.3. This sensor was attached to the EyeBot via the digital I/O pins of it. The signal could be obtained by using another RoBIOS function. The function reads the entire low-level I/O latches, and the user has to mask out the bit of the desired channel to acquire the actual value. However, these sensors were only needed while the calibration of the legs was in progress. A function was created, which comprises the whole calibrating process. This function is called before starting the interrupt function (see Chapter 8). For an accurate calibration, each leg was moved slowly and separately and the optical sensor was monitored. When the slot in the crank passes through the sensor, the encoder counters were reset. The exact position of the legs was now available at any given time.

For a static balance control of the robot, the exact position of the feet was crucial. The feet position had to be kept before the moving mass system could be controlled to change the total COM of the system. To keep the exact position of the feet, the controller for the feet, described in chapter 5 was fed with the desired value 0 and the actual position of the feet was read out. Supplying the motors with the controller-calculated PWM signal, the motor provides just enough torque to maintain the actual position.

6.2.2. Inclinometer feedback

In Chapter 4.3 the purpose of the inclinometers was explained. Several difficulties were encountered because the inclinometers are attached to the analog input of the EyeBot. To obtain a reliable value of the inputs while changing the channel, a certain latency time had to be implemented. A function was created which returns the value of the desired input. The function waits the required time, before the value is read and returned. This is realized by reading the input more than once and doing this until it is ensured that the ADC delivers the right value.

For a better overview of the actual position, using the inclinometer values, the actual pitch and roll angle was displayed on the display of the EyeBot. The angle was calculated in a function especially programmed for this, because the inclinometers only provided an analog signal, which was stored in a variable. This routine calculates the middle of measurement range, estimates this as 0° , and multiplies this value with the digit to degree ratio of the inclinometer. Until the inclinometer was not fixed rigidly to the robot, a calibrating function was programmed to determine the point where the angle was exactly 0° . Furthermore, a graphical representation in form of a slider, which moves from side to side according to the actual angle of the robot, was shown on the display to have a coarse overview of the value.

6.2.3. Position feedback of the moving mass

To have the actual position of the moving mass, the encoder for this motor was used. As mentioned before, these sensors are incremental and not absolute. Hence, a function was programmed to calibrate the moving mass before starting the control, because no sensors were attached at the limiters, which could provide the information that the counterweight has arrived at the maximum position. The function was called before the interrupt function was started (see Chapter 8). In this function, the moving mass is moved slowly to one end and the actual speed is computed. If the actual speed is zero, the mass has arrived at the limiter. Then the counter is reset and the mass is moved to the other end until the speed is again zero. Finally, the actual value of the counter is divided by two and this value is the centre of the total movement. The mass is moved to this position and the counter is reset again. At this moment the mass is located in the middle, the counter is zero and the function exits returning the value of the maximal movement in both directions.

Now a closed loop control for the mass system was easy to implement, since the system was symmetric and the zero value of the counter was definitely at the centre. The actual position was also displayed on the EyeBot screen using a slider with a cross to represent the mass. This offers a better overview for the user and the certainty that the real position coincides with the calculated position.

6.3. Implementation of close loop control for static balance

After having initialized and calibrated the various parts of the robot, a proper control algorithm could be designed and implemented. For the foot movement the control was already finished and only the desired had to be set to zero. The position was read and this position was maintained using the control algorithm for each leg. The initial position of the feet was irrelevant, because after once having calibrated the feet, the position is refreshed instantly before the controller starts.

For the moving mass system, a controller had to be added in order to use the values of the inclinometer that supplies the roll of the biped robot. As the values of the inclinometers are analog, they can be easily stored and processed.

Two functions were added to the project concerning the moving mass. The first function `WeightMove` is for positioning, and can be used by any other function just to move the mass to a defined position. The function needs the position to move to, the desired speed to move at, and the maximum positions to which the mass can be moved in both directions (this value is provided by the calibration function of the moving mass). The function drives at the desired speed toward the desired point and while getting close to this point, the speed is reduced until arriving at this point. This function is a compromise between speed and accuracy. If the time to move to a certain position has to be short, the speed has to be high. However, the faster the mass is, the harder it is to achieve precisely the desired position, because of the high kinetic energy the mass possesses. Thus, the speed is also an input of the function.

The other function is the controller for the moving mass system. For a static balance, a simpler controller was conceptualized, for the reason that the movement did not need to be as fast as for dynamic control. The accurate position of the mass was also not crucial, because every movement of the mass produced oscillations on the entire system and this was undesired. Hence, every needless movement was avoided and only if the system was getting unstable, a movement was taken into consideration. The easiest way to realize such a control was to build up a set of rules that look quite similar to set of fuzzy rules. As shown in example Code 6.1, depending on the actual angle of the system, the function

`WeightMove` is called with different parameters. For small angles ($<2^\circ$) the controller does nothing in order to avoid an unnecessary movement and resulting vibrations. The parameters needed for this function are the actual angle of roll and the maximum position in which the mass can be moved in both directions. The second value is needed to avoid the movement against the limiter. If the mass has reached the limiter, the motor will not try to move further on.

```
void WeightControl (double angle_lr, int max)
{
    // rules for weight movement for static balance

    if (angle_lr > 2.0 && angle_lr <3)
        WeightMove (QUADRead(qH[2]) - 150 , max, 50);
    if (angle_lr > 3 && angle_lr <5)
        WeightMove (QUADRead(qH[2]) - 500 , max, 70);
    if (angle_lr > 5 && angle_lr <7)
        WeightMove (QUADRead(qH[2]) - 1000, max, 80);
    if (angle_lr > 7)
        WeightMove (QUADRead(qH[2]) - 2000, max, 100);

    if (angle_lr < -2.0 && angle_lr > -3)
        WeightMove (QUADRead(qH[2]) + 150 , max, 50);
    if (angle_lr < -3 && angle_lr > -5)
        WeightMove (QUADRead(qH[2]) + 500 , max, 70);
    if (angle_lr < -5 && angle_lr > -7)
        WeightMove (QUADRead(qH[2]) + 1000, max, 80);
    if (angle_lr < -7)
        WeightMove (QUADRead(qH[2]) + 2000, max, 100);
}
```

Code 6.1: Rules for weight movement for static balance

After implementing these functions in the project, the controller function for the mass was added into the interrupt routine (see Chapter 8). The result was a very stable balancing of the robot, once the different parameters were adapted and timing problems were overcome. The robot can balance on different inclined surfaces regardless of the feet actual position. Even if an external force is acting on it, the robot is able to keep balance until a certain point. Slow changes of the forces acting on the system can be compensated very well, though for very fast changes, the controller still has to be improved.

7. Dynamic balancing control

7.1. Zero moment point

An often-used stability criterion for dynamic balancing is the usage of the ZMP. As mentioned before, it is the point on the ground where the sum of all moments is equal to zero. The ZMP can be calculated knowing the gravity of the whole system and the weight, position, and the acceleration vector of the different point-masses [12]. If the acceleration equals zero, the ZMP has the same position as the NPCM. As it is for a stable balance with the COM, the ZMP is constrained to lie within the ground support polygon to avoid the robot falling while moving. While both feet are in contact with the ground, the polygon is determined by the outer corners of the feet. During the single support phase of a walking gait, the support polygon is provided only by the one foot with ground contact. As the system becomes unstable, this point will lie on the boundary of the support polygon. While performing a dynamic walk, this point can lie outside these boundaries for shorter periods, but has to return into a stable position to restore the balance of the system.

7.2. Design of a close loop control for a walking gait

As realized in chapter 7.1, the ZMP provides enough information to build a close loop control for the robot walking gait. However, the calculation of the ZMP is very computationally intense. Thus, a real-time computation on the controller itself is beyond question. This would make the movement too slow and the computation time would nearly be totally consumed by the algorithm. As done before for the COM, using the relationship between the angle and the position of the COM, a sensor like an accelerometer could be used to watch the acceleration of the COM and approximate the actual position of the ZMP.

At this stage of development, the robot was not tested under these conditions, because of time limitation. In addition, an accelerometer was not tested on this robot. Nevertheless, the next step to go would be to implement another sensor and investigate the relationship between the actual acceleration supplied by this sensor and the position of the ZMP. If this is established, an appropriate control algorithm could be designed using the same methods applied for the static balance. Finally, the design of the moving mass system could be adapted, in order to provide the system with another DOF, and the designed controller could be implemented in the existing software architecture.

8. Software architecture

In this chapter a brief overview of the software functions and the system architecture is given. From the flow charts, the calling sequence and the relationship between the functions are visualized.

8.1. System components

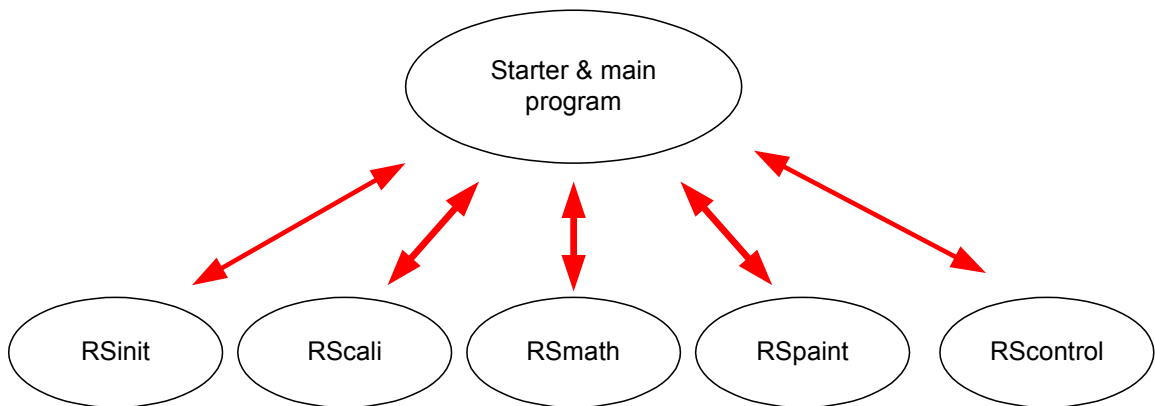


Figure 8.1: Diagram of the project structure

As Figure 8.1 shows, there are six different main software modules. Up to now, the different modules contain the functions that can be called from any other function. Error management is also included in each function to intercept any errors during these phases and report these to the user, using the display as an interface. In the future, it is better to embed these functions in classes, in order to achieve a better protection of the private segments of each function and achieve a better interaction between the different modules.

The RSinit module comprises functions to initialize and release the motors and the encoders. RScali provides functions to calibrate the different elements of the robot such as the legs, the inclinometers and the moving mass system. The RSmath module includes additional functions for specific calculations such as round, limit or even a fast sinus and cosines calculation using a look up table. RSpaint offers several supplementary functions for displaying objects like circles or crosses on the LCD of the EyeBot.

8.2. Diagram of the process structure

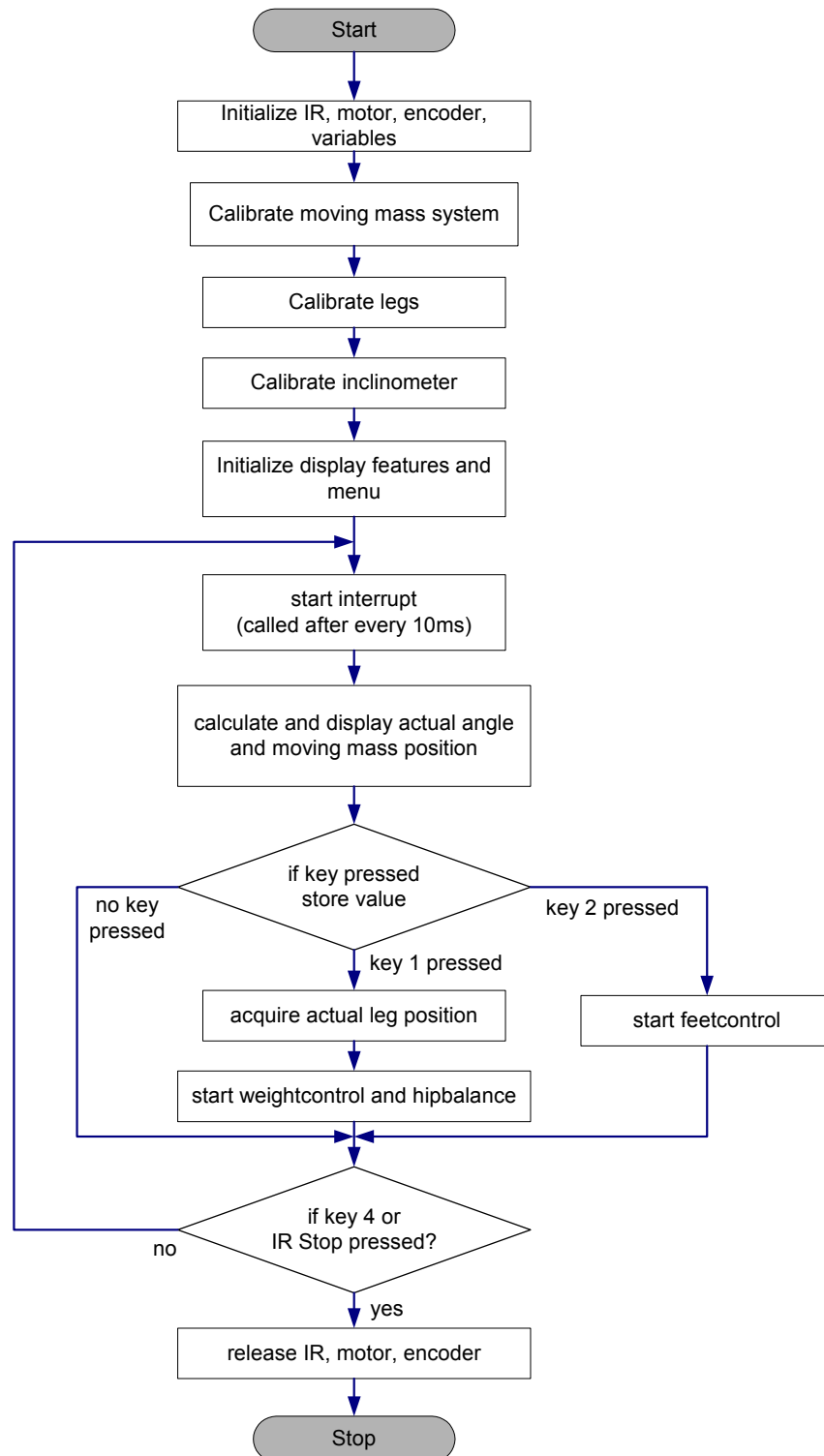


Figure 8.2: Flowchart of the actual process structure

Figure 8.2 shows the actual program sequence, with two independent controller-applications that the user can choose between by using either the soft buttons or the IR control unit. First, all needed systems are initialized and calibrated. After this, the interrupt function is started. The important values are displayed on the screen and refreshed every 10ms. The user can choose between the leg movement and the static balance control. The leg movement only uses the leg controller at a desired speed of 80%. This application is to be upgraded, because in the future it should comprise the dynamic control, which means an additional controller for the moving mass system. The static balance control, as mentioned in Chapter 6, uses the leg controller and another controller for the moving mass system.

8.3. Diagram of the leg control structure

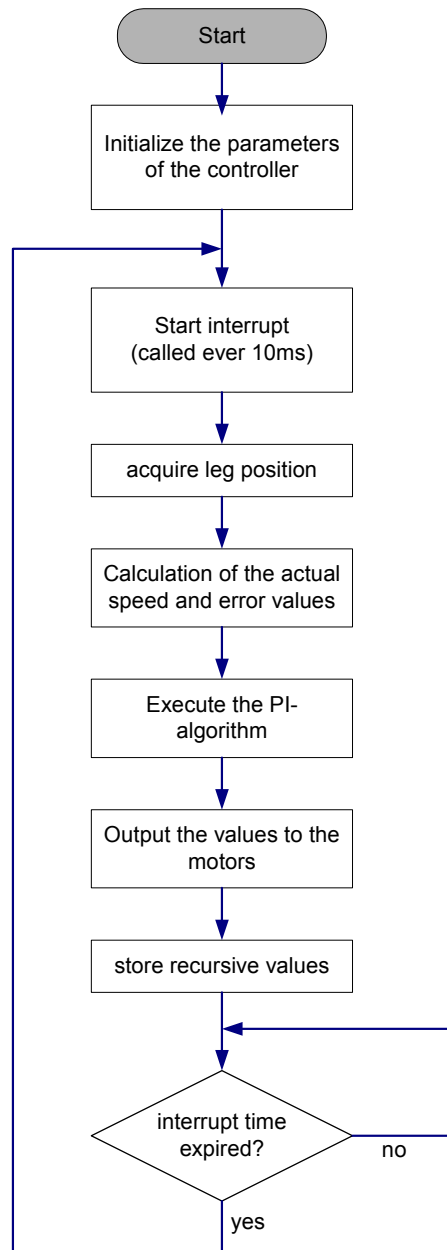


Figure 8.3: Flowchart of the leg control algorithm

To give a better overview of the leg control algorithm, Figure 8.3 shows the detailed sequence of the executions of the different elements. This chart is just a general idea of how the sample algorithm is integrated in the system. The actual sequence varies a little, because other supplementary functions have been added. The added functions have been omitted in Figure 8.3 in order to minimise complexity.

9. Evaluation and future work

The designed close loop control algorithm for the leg movement is efficient and fast. The result is satisfying and the algorithm does not take much computation time, so that enough time is available for other applications. The controller for the moving mass system for the static balance also is kept manageable and it seems to perform better than expected. At this point, for the static balance of the robot only a few adaptations had to be done, in order to improve the control.

Concerning the dynamic control, there was not enough time to design such a software control algorithm, because a lot of time was spent not only on the development of the mechanical part, but also designing and testing the different sensors. However, using the current mechanical design this control would not properly work. This means that further improvements have to be done regarding the moving mass system and the foot design. Having only one DOF for the moving mass would cause many problems if the system tends to fall forwards or backward while performing a walking gait.

With references to the foot design, this could be improved by using a spring for returning the foot to a reference position, during the period where the leg is not in contact with the ground. This could also be realised by placing another rubber block in front of the ankle joint, limiting the movement of the passive foot.

Weight reduction is also a point to keep in mind while improving the system. The weight of the current design could be reduced without losing the stiffness and robustness of the frame design by removing excess material.

The filter circuit of the analog sensors also needs to be improved, since the software control of the robot is a sampling system and aliasing has to be avoided. At this point, a capacitor is connected at the output of the inclinometers as a simple low pass. Nevertheless, in order to ensure that the right value is read by the system a more complex filter has to be added, whose frequency is at least the double the sampling frequency of the controller.

Until now, the structure of the software was kept as simple as possible, because the hardware changed several times and the program had to be adapted every time. As future work, the software part has to be enhanced. The functions have to be encapsulated in classes and additional error management has to be included.

10. Conclusion

In this thesis, I presented the close loop control I developed for a biped robot with minimal number of actuators. At the moment, the project is still not finished and different elements still need to be changed and improved. For static balance, the control was finished, but taking into account the development time of the mechanical part, the dynamic control and finally the dynamic walking could not be finished during my research.

Several methods from different fields of electrical engineering, such as control engineering, measurement engineering or computer sciences were applied successfully to achieve the attained results. First results showed that the principle of this walking machine is by far not as previously estimated, and gives confidence that this robot can one day perform a walking gait. Thus, the success of the project will lie in the hands of whoever will use my research and resolves to continue the studies and apply the changes I suggested.

11. Appendices

Appendix A: Figure Index and sources

Figure 2.1: The three anatomic planes: frontal, sagittal and transversal	3
Figure 2.2: Leg position during one-half cycle.....	4
Figure 2.3: The cyclic phase rotation of biped walking	4
Figure 2.4: Side view of the movement of the legs during a walking gait	5
Figure 2.5: The humanoid robot ASIMO	8
Figure 2.6: Planar One-Leg Hopper from the MIT (Raibert, M. H. 1980-82)	9
Figure 2.7: 3D One-Leg Hopper from the MIT (Raibert, M. H. 1983-84).....	9
Figure 2.8: Walking bipeds Johnny walker, Jack Daniels and Andy	10
Figure 2.9: Different types of air-muscles	12
Figure 3.1: Different type of sensors: Inclinometer a), Gyroscope b), Accelerometer c), Pressure Sensor d).....	13
Figure 3.2: Schematic illustration of the leg mechanism.....	15
Figure 3.3: Leg mechanism	16
Figure 3.4: First prototype of the moving mass system.....	17
Figure 3.5: Schematic illustration of the second prototype of the moving mass system...18	
Figure 3.6: Moving mass system of the third prototype	19
Figure 3.7: Side view of the sliding bed and the moving mass	20
Figure 3.8: Actual design of the moving mass system	20
Figure 3.9: Curved foot prototype	21
Figure 3.10: Actual passive foot	21
Figure 3.11: Prototype 1	22
Figure 3.12: Prototype 2	22
Figure 3.13: Prototype 3	23
Figure 3.14: Prototype 4	23
Figure 4.1: The EyeBot controller	24
Figure 4.2: Faulhaber DC Motor2224R006S with inbuilt gearbox and encoders	25
Figure 4.3: PWM signal at different values	26
Figure 4.4: PCB with optical sensor	28

Figure 4.5: Side plate of one leg with slotted crank and optical sensor PCB.....	28
Figure 4.6: Inclinometer	29
Figure 5.1: Block diagram of the two legs	36
Figure 5.2: Step response at 80% of the PWM with tangent method parameters	39
Figure 5.3: Step-function, step-response and approximation of it using WinFACT.....	41
Figure 5.4: Block diagram of the open loop system.....	44
Figure 5.5: Block diagram of the close loop control of the system.....	45
Figure 5.6: Step response using a close loop control with activated disturbance.....	47
Figure 5.7: Step response using the optimized close loop control with activated disturbance.....	48
Figure 6.1: Graph of the frontal mass distribution and COM position (scale 1:2).....	53
Figure 6.2: Graph of the sagittal mass distribution and COM position (scale 1:2).....	54
Figure 8.1: Diagram of the project structure	63
Figure 8.2: Flowchart of the actual process structure.....	65
Figure 8.3: Flowchart of the leg control algorithm	67

Appendix B: References

- [1] **Kamphausen, Jörg;** *Zweibeiniges Gehen, Seminar Roboter im Alltag*, Department of computer sciences, University of Dortmund, Germany, Available from (15.04.03):
<http://ls1-www.cs.uni-dortmund.de/~asg/asg/Paper/SeminarRIA02/kamphausen.ausarbeitung.pdf>
- [2] **Dienelt, Martin;** *Neuere Entwicklungen auf dem Gebiet humanoider Roboter, Wie humanoide Roboter laufen*; Department of Robotics, University of München, Germany, Available from (15.04.03):
http://www.siegert.informatik.tu-muenchen.de/lehre/seminare/hs_ss02/WalkingRobots.pdf
- [3] **L.Kun, Andrew;** *A sensory-based adaptive walking control algorithm for variable speed biped robot gaits*, University of New Hampshire, England 1997
- [4] **Humanoid Robot ASIMO;** Honda Motor Co, Available from (15.04.03):
<http://www.honda-p3.com/english/html/asimo/frameset2.html>
- [5] **Massachusetts Institute of Technology (MIT) Leg laboratory,** Available from (15.04.03):
<http://www.ai.mit.edu/projects/leglab/robots/robots-main.html>
- [6] **Mobile Robot Lab,** Department of electrical and electronic engineering, University of Western Australia, Australia, Available from (15.04.03):
<http://robotics.ee.uwa.edu.au/>
- [7] **Air Muscles,** Shadow Robot Company Ltd., Available from (15.04.03):
<http://www.shadow.org.uk/products/airmuscles.shtml>
- [8] **Jungpakdee, Kitirat;** *Design and construction of a minimal biped walking mechanism*, School of Mechanical Engineering, University of Western Australia, Australia, 2002
- [9] **Lutz, H.;Wendt W.;** *Taschenbuch der Regelungstechnik*, Publisher Harri Deutsch (2002)
- [10] **Ogata, Katsuhiko;** *Modern Control Engineering*, Prentice Hall (1990)
- [11] **Hardt, Michael; von Stryk, Oskar;** *Increasing stability in dynamic gaits using numerical optimization*, Simulation and Systems Optimization Group, University of Darmstadt, Germany, Available from (17.04.03):
<http://www.sim.informatik.tu-darmstadt.de/publ/download/2002-ifac-hardt-vonstryk.pdf>
- [12] **Streuer, Manuel;** *Vision Guided Virtual Walking Machine*, Department of Robotics, University of München, Germany, Available from (17.04.03):
http://www.siegert.informatik.tu-muenchen.de/lehre/seminare/hs_ws0203/steuerer_final.pdf

Appendix C: Faulhaber Specification Sheets (extract)



DC-Micromotors

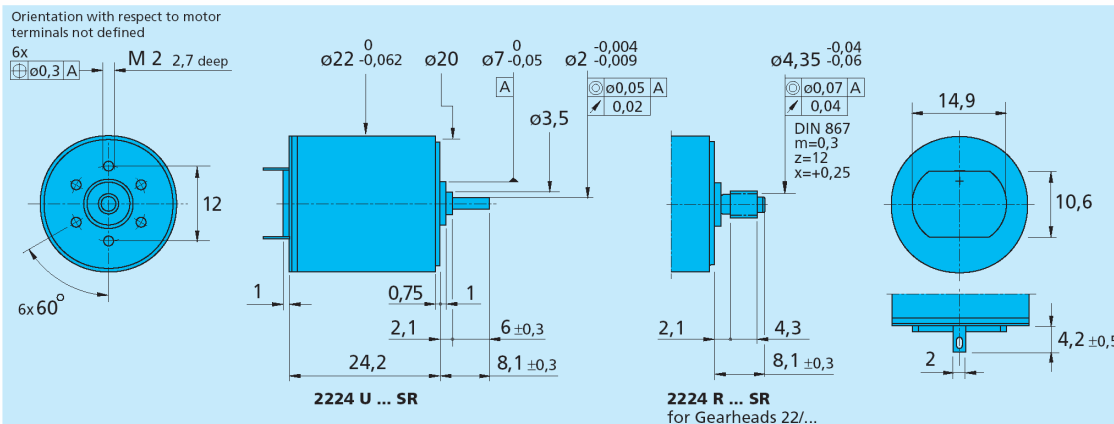
5 mNm

Precious Metal Commutation

For combination with
Gearheads:
20/1, 22E, 22/2, 22/5, 22/6, 23/1, 38/3
Encoders:
IE2

Series 2224 ... SR

	2224 U	003 SR	006 SR	012 SR	018 SR	024 SR	036 SR	
1 Nominal voltage	U_N	3	6	12	18	24	36	Volt
2 Terminal resistance	R	0,56	1,94	8,71	17,50	36,30	91,40	Ω
3 Output power	$P_2 \text{ max.}$	3,92	4,55	4,05	4,54	3,88	3,46	W
4 Efficiency	$\eta \text{ max.}$	80	82	82	82	81	80	%
5 No-load speed	n_o	8 100	8 200	7 800	8 100	7 800	7 800	rpm
6 No-load current (with shaft ϕ 2,0 mm)	I_o	0,066	0,029	0,014	0,010	0,007	0,005	A
7 Stall torque	M_H	18,5	21,2	19,8	21,4	19,0	16,9	mNm
8 Friction torque	M_R	0,23	0,2	0,2	0,21	0,2	0,22	mNm
9 Speed constant	k_n	2 730	1 380	657	454	328	219	rpm/V
10 Back-EMF constant	k_E	0,366	0,725	1,520	2,200	3,040	4,560	mV/rpm
11 Torque constant	k_M	3,49	6,92	14,50	21,00	29,10	43,50	mNm/A
12 Current constant	k_I	0,286	0,144	0,069	0,048	0,034	0,023	A/mNm
13 Slope of n-M curve	$\Delta n / \Delta M$	438	387	394	379	411	462	rpm/mNm
14 Rotor inductance	L	11	45	200	450	800	1 800	μ H
15 Mechanical time constant	τ_m	11	11	11	11	11	11	ms
16 Rotor inertia	J	2,4	2,7	2,7	2,8	2,6	2,3	gcm ²
17 Angular acceleration	$\alpha \text{ max.}$	77	78	74	77	74	74	$\cdot 10^3 \text{ rad/s}^2$
18 Thermal resistance	$R_{th 1} / R_{th 2}$	5 / 20						K/W
19 Thermal time constant	τ_{w1} / τ_{w2}	6,8 / 440						s
20 Operating temperature range:								
- motor		- 30 ... + 85 (optional - 55 ... + 125)						$^{\circ}$ C
- rotor, max. permissible		+ 125						$^{\circ}$ C
21 Shaft bearings		sintered bronze sleeves		ball bearings		ball bearings, preloaded		
22 Shaft load max.:		(standard)		(optional)		(optional)		
- with shaft diameter		2,0		2,0		2,0		mm
- radial at 3000 rpm (3 mm from bearing)		1,5		8		8		N
- axial at 3000 rpm		0,2		0,8		0,8		N
- axial at standstill		20		10		10		N
23 Shaft play:								
- radial	\leq	0,03		0,015		0,015		mm
- axial	\leq	0,2		0,2		0		mm
24 Housing material		steel, black coated						
25 Weight		46						g
26 Direction of rotation		clockwise, viewed from the front face						
Recommended values								
27 Speed up to	$n_e \text{ max.}$	8 000	8 000	8 000	8 000	8 000	8 000	rpm
28 Torque up to	$M_e \text{ max.}$	5	5	5	5	5	5	mNm
29 Current up to (thermal limits)	$I_e \text{ max.}$	2,200	1,200	0,570	0,400	0,280	0,180	A



For notes on technical data and lifetime performance refer to "Technical Information".

For options on DC-Micromotors refer to page 62. Specifications subject to change without notice.



Encoders

Magnetic Encoders

Features:
 64 to 512 Lines per revolution
 2 Channels
 Digital output

Series IE2 – 512

		IE2 – 64	IE2 – 128	IE2 – 256	IE2 – 512	
Lines per revolution	N	64	128	256	512	
Signal output, square wave		2				channels
Supply voltage	V _{DD}	4,5 ... 5,5				V DC
Current consumption, typical (V _{DD} = 5 V DC)	I _{DD}	typ. 6, max. 12				mA
Output current, max. ¹⁾	I _{OUT}	5				mA
Pulse width	P	180 ± 45				°e
Phase shift, channel A to B	Φ	90 ± 45				°e
Signal rise/fall time, max. (C _{LOAD} = 50 pF)	tr/tf	0,1 / 0,1				µs
Frequency range ²⁾ , up to	f	20	40	80	160	kHz
Inertia of code disc	J	0,09				gcm ²
Operating temperature range		- 25 ... + 85				°C

¹⁾ V_{DD} = 5 V DC: Low logic level < 0,5 V, high logic level > 4,5 V: CMOS and TTL compatible

²⁾ Velocity (rpm) = f (Hz) x 60/N

Ordering information

Encoder	number of channels	lines per revolution	in combination with:
IE2 – 64	2	64	DC-Micromotors series 1336 ... C, 1516 ... SR, 1524 ... SR, 1717 ... SR, 1724 ... SR, 1727 ... C, 2224 ... SR, 2342 ... CR, 2642 ... CR, 2657 ... CR, 3242 ... CR, 3257 ... CR, 3863 ... C brushless DC-Servomotors series 1628 ... B, 2036 ... B, 2444 ... B
IE2 – 128	2	128	
IE2 – 256	2	256	
IE2 – 512	2	512	

Features

These incremental shaft encoders in combination with the FAULHABER DC-Micromotors and brushless DC-Servomotors are used for indication and control of both, shaft velocity and direction of rotation as well as for positioning.

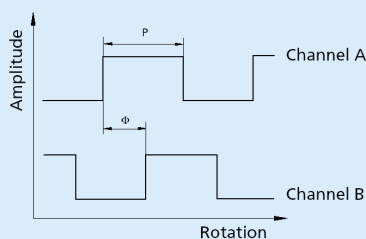
The encoder is integrated in the DC-Micromotors SR-Series and extends the overall length by only 1,4 mm and build-up option for DC-Micromotors and brushless DC-Servomotors.

Hybrid circuits with sensors and a low inertia magnetic disc provide two channels with 90° phase shift.

The supply voltage for the encoder and the DC-Micromotor as well as the two channel output signals are interfaced through a ribbon cable with connector.

Details for the DC-Micromotors and suitable reduction gearheads are on separate catalog pages.

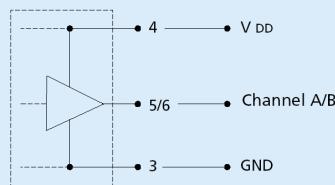
Output signals / Circuit diagram / Connector information



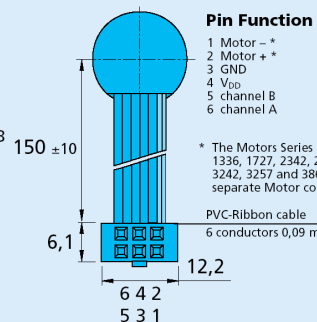
Admissible deviation of phase shift:

$$\Delta\Phi = \left| 90^\circ - \frac{\Phi}{P} * 180^\circ \right| \leq 45^\circ$$

Output signals
 with clockwise rotation as seen from the shaft end



Output circuit
 Note: Motor terminal resistance increases by approx. 0,4 Ω



Pin Function

- 1 Motor – *
- 2 Motor + *
- 3 GND
- 4 V_{DD}
- 5 channel B
- 6 channel A

* The Motors Series 1336, 1727, 2342, 2642, 2657, 3242, 3257 and 3863 have separate Motor connector.

PVC-Ribbon cable
 6 conductors 0,09 mm²

Connector
 DIN-41651
 grid 2,54 mm

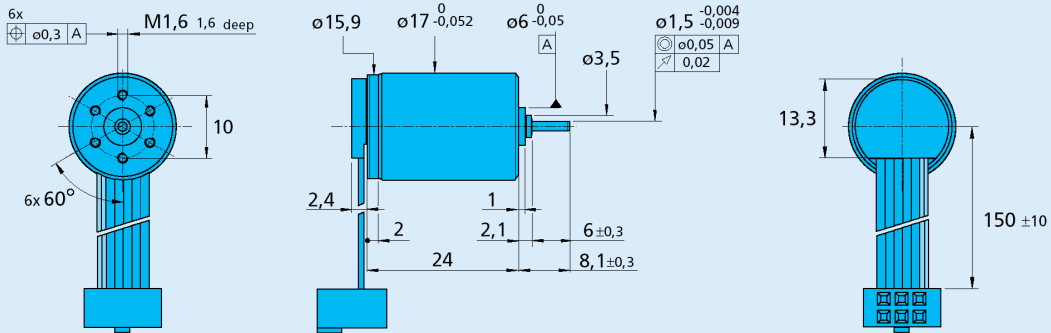
For notes on technical data and lifetime performance refer to "Technical Information".

Specifications subject to change without notice.



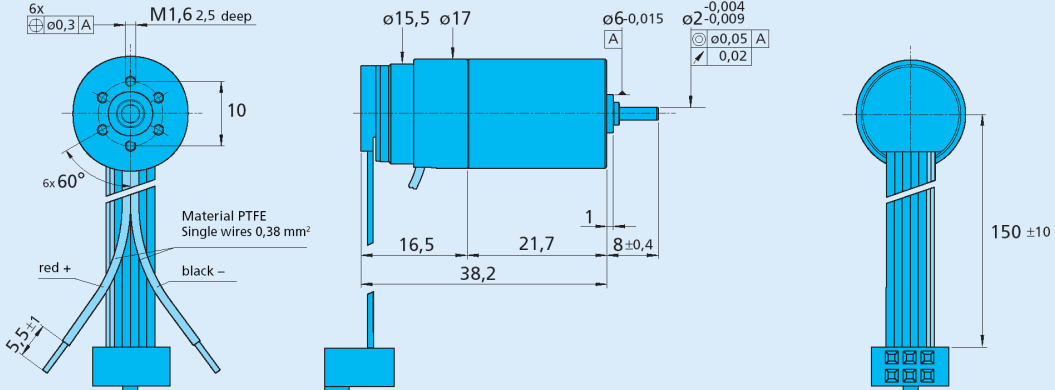
DC-Micromotor 1724 T ... SR with Encoder IE2 – 16 ... 512

Orientation with respect to cable not defined



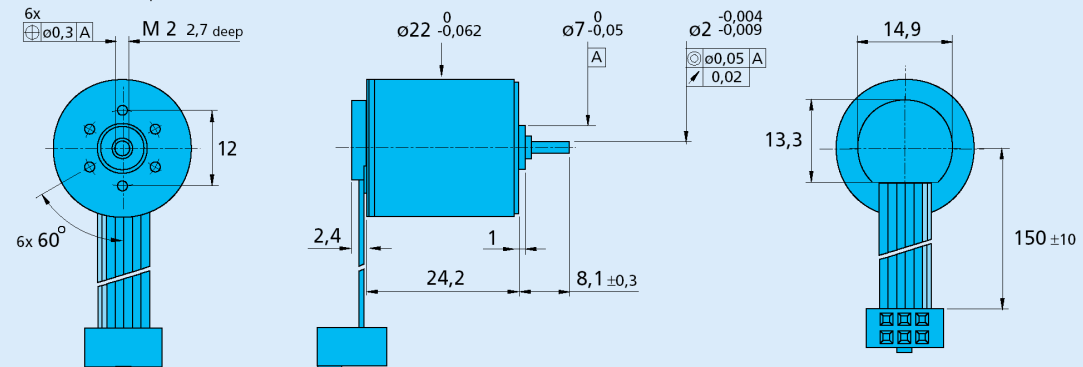
DC-Micromotor 1727 U ... C - 123 with Encoder IE2 – 64 ... 512

Orientation with respect to cable not defined



DC-Micromotor 2224 U ... SR with Encoder IE2 – 16 ... 512

Orientation with respect to cable not defined



For notes on technical data and lifetime performance refer to "Technical Information".

Specifications subject to change without notice.

Appendix D: Seika Inclinometer Data Sheets

www.seika.de

N2, N3, N4



Small Inclinometers for inclination measurement in the ranges of ± 10 , ± 30 and ± 70 degrees.

Features

- linear output characteristics
- high measurement accuracy
- high long-term stability
- hysteresis free output signal
- minimal zero point drift
- integrated sensor electronics
- low power consumption
- small housing
- light weight
- different output signal options
- no interference by ambient electromagnetic fields
- shockproof as without moving mechanical parts
- hermetically sealed
- sensor electrically isolated from point of measurement using high quality plastic housing – no ground connections
- zero point adjustable through 360° using clamping ring

Description

The N2, N3 and N4 are capacitive, liquid based inclinometers with integrated sensor electronics. They are manufactured either with an analog DC or a pulse width modulated output. The sensor electronics require only minimal power and are in conjunction with the capacitive primary transformer characterized by high accuracy, a high signal-to-noise ratio and high long-term stability.

The measurement technique enables a linear relationship between the angle to be measured and the output signal. The determined angle is independent of the local gravitational acceleration, that means that no matter where the measurement is being taken, whether in Europe, Australia, on Mount Everest or on the moon, the inclination will be measured correctly anywhere.

Application

The inclinometers N2, N3 and N4 are suitable for applications requiring a small, light sensor for measurement of relatively large inclinations.

Typical areas of application include measuring instruments and inspection systems, vehicles, automation and safety engineering, scientific devices, medical and communications equipment as well as navigational systems.

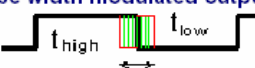
SEIKA Mikrosystemtechnik GmbH * Ellharter Str.10 * D-87435 Kempten * Tel: 0831-25532 Fax: 0831-25534
Internet: <http://www.seika.de> * <http://www.seika.net> * Email: seika@seika.de

(7/2001) - NS 3 -

www.seika.de

N2, N3, N4

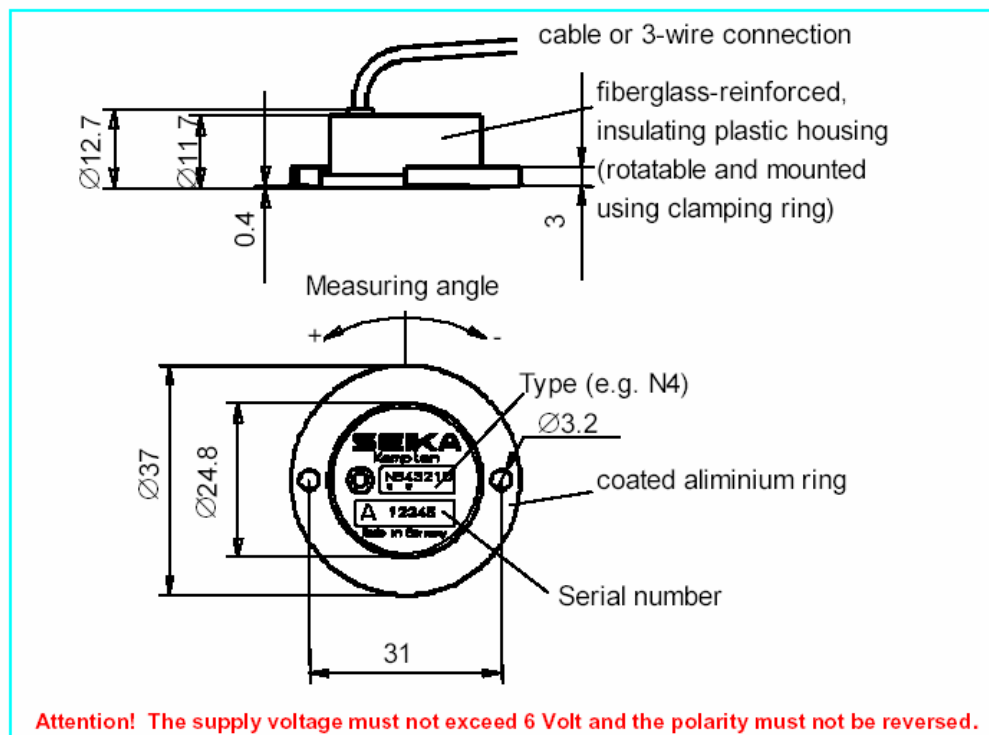
Technical Specifications

Type	N2	N3	N4
Dimensions	see dimension drawing		
Measuring range	±10 degrees	±30 degrees	±70 degrees
Resolution	<0,002 degrees	<0,005 degrees	<0,01 degrees
Linearity deviation	<0,2% of measuring range		
Transverse sensitivity	<1% at 30° tilt		
Settling time	<0,3 seconds		
Supply voltage U_b (regulated)	5 Volt		
Permissible supply voltage range U_{bz}	3...6 Volt		
Current drawn at $U_h = 5V$	approx. 1mA		
Degree of protection	IP65		
Operating temperature	-40...+85°C		
Storage temperature	-45...+90°C		
Weight (without clamping ring or cable)	18,5 grams		
Electrical connection	3 highly flexible wires Øapprox. 1mm, length 18cm optional: 0,5m shielded cable Ø2,1mm		
Values for analog DC output at $U_{bN}=5V$			
Sensitivity	approx. 12 mV/degree	approx. 6 mV/degree	approx. 3.5 mV/degree
Temperature drift of sensitivity	-0,17% / K	<-0,12% / K	
Temperature drift of zero point	<±0,05mV/K	<±0,025mV/K	
Zero offset at $U_b=5V$	2,5±0,1Volt - generally: $0,5U_b \pm 4\%$		
Output impedance	10kΩ		
Values for pulse width modulated output at $U_{bN}=5V$			
			
Sensitivity at zero point $dt_{EJ}/(t_{high}+t_{low})$	approx. $76 \cdot 10^{-3}/deg.$	approx. $33 \cdot 10^{-3}/deg.$	approx. $20 \cdot 10^{-3}/deg.$
Temperature drift of sensitivity	-0,17% / K	<-0,12% / K	
Temperature drift of zero point	<±1,6*10 ⁻⁴ F.S./K	<±8*10 ⁻⁵ F.S./K	
Zero pulse width ratio t_{high}/t_{low}	1±4%		
Output frequency	approx. 20Hz to approx. 1MHz (optional)		

www.seika.de

N2, N3, N4

Dimensions and Connections

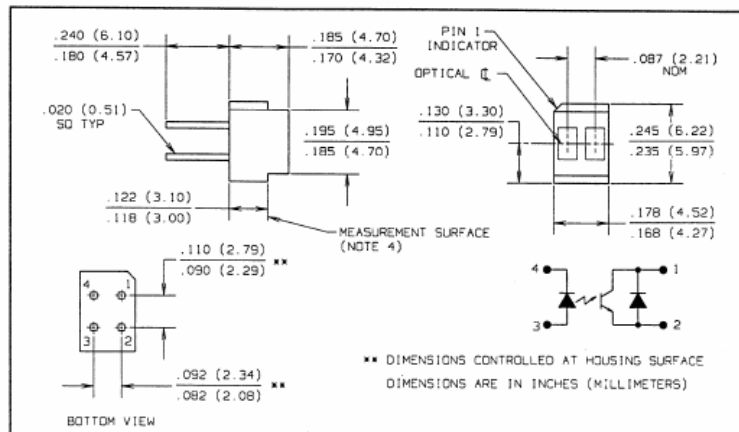
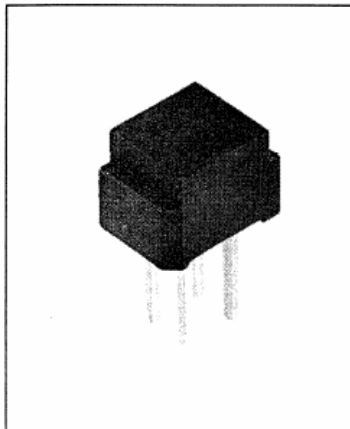


Appendix E: Reflective object sensor Data Sheet



Product Bulletin OPB608
June 1996

Reflective Object Sensors Types OPB608A, OPB608B, OPB608C



Features

- Phototransistor output
- Unfocused for sensing diffuse surface
- Low cost plastic housing
- Enhanced signal to noise ratio
- Reduced ambient light sensitivity

Description

The OPB608 consists of an infrared emitting diode and an NPN silicon phototransistor mounted "side-by-side" on parallel axes in a black opaque plastic housing. Both the emitting diode and phototransistor are encapsulated in a filtering epoxy to further reduce ambient light noise. The phototransistor responds to radiation from the emitter only when a reflective object passes within its field of view. The phototransistor has enhanced low current roll off to improve the contrast ratio and immunity to background irradiance.

Absolute Maximum Ratings (T_A = 25° C unless otherwise noted)

Storage and Operating Temperature -40° C to +85° C
Lead Soldering Temperature [1/16 inch (1.6 mm) from case for 5 sec. with soldering iron]..... 240° C⁽¹⁾

Input Diode

Forward DC Current 50 mA
Peak Forward Current (1 μs pulse width, 300 pps) 3.0 A
Reverse DC Voltage 2.0 V
Power Dissipation 75 mW⁽²⁾

Output Phototransistor

Collector-Emitter Voltage 30 V
EmitterReverse Current 10 mA
Collector DC Current 25 mA
Power Dissipation 100 mW⁽³⁾

Notes:

- (1) RMA flux is recommended. Max 20 grams force may be applied to the leads when soldering. Duration can be extended to 10 sec. max when flow soldering.
- (2) Derate linearly 1.25 mW/° C above 25° C.
- (3) Derate linearly 1.67 mW/° C above 25° C.
- (4) d is the distance from the assembly measurement surface to the reflective surface.
- (5) Measured using Eastman Kodak neutral white test card with 90% diffuse reflectance as a reflecting surface.
- (6) Off state collector current I_{C(OFF)} is measured with no reflective surface in the optical path.

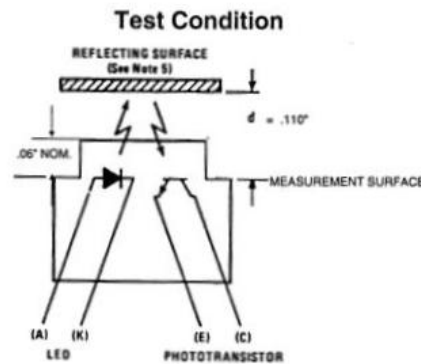
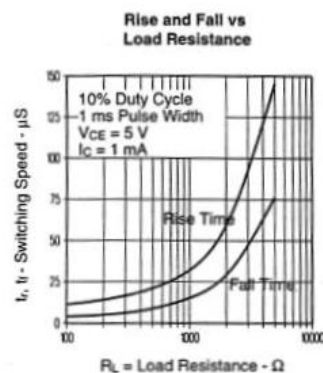
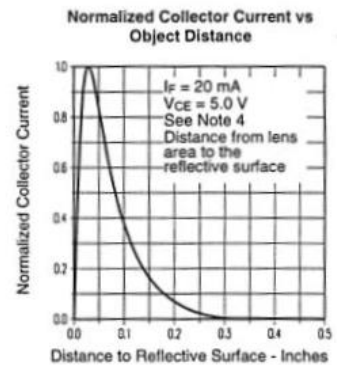
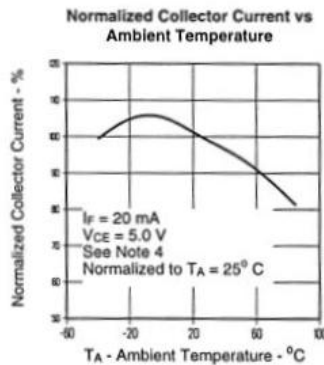
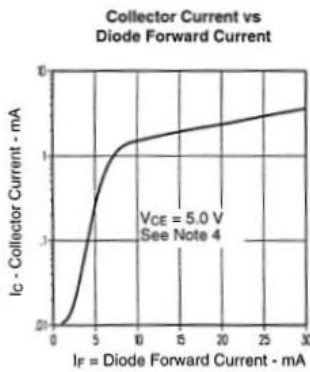
Types OPB608A, OPB608B, OPB608C

Electrical Characteristics ($T_A = 25^\circ\text{C}$ unless otherwise noted)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
Input Diode					
V_F	Forward Voltage		1.70	V	$I_F = 20\text{ mA}$
I_R	Reverse Current		100	μA	$V_R = 2.0\text{ V}$
Output Phototransistor					
$V_{(BR)CEO}$	Collector-Emitter Breakdown Voltage	30		V	$I_C = 100\ \mu\text{A}$
I_{ECO}	Emitter Reverse Current		100	μA	$V_{EC} = 0.4\text{ V}$
I_{CEO}	Collector Dark Current		100	nA	$V_{CE} = 5.0\text{ V}, I_F = 0, E_\theta = \leq 0.10\ \mu\text{W}/\text{cm}^2$
Combined					
$I_{C(ON)}$	On-State Collector Current	OPB608A OPB608B OPB608C	2.0 1.0 0.5	4.0 mA mA mA	$V_{CE} = 5.0\text{ V}, I_F = 20\text{ mA}, d = 0.110\text{ in. (2.79 mm)}^{(4)(5)}$
$I_{C(OFF)}$	Off-State Collector Current		100	nA	$V_{CE} = 5.0\text{ V}, I_F = 20\text{ mA}^{(6)}$

REFLECTIVE OBJECT SENSORS

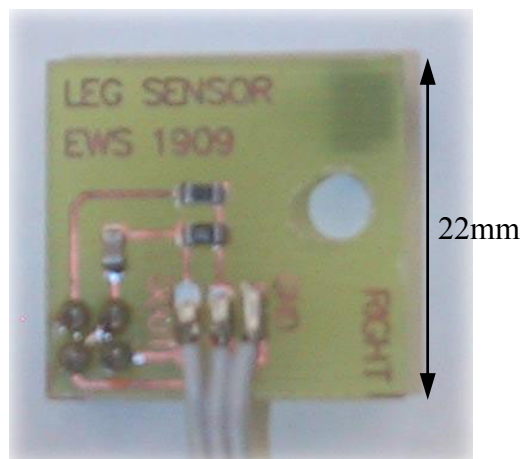
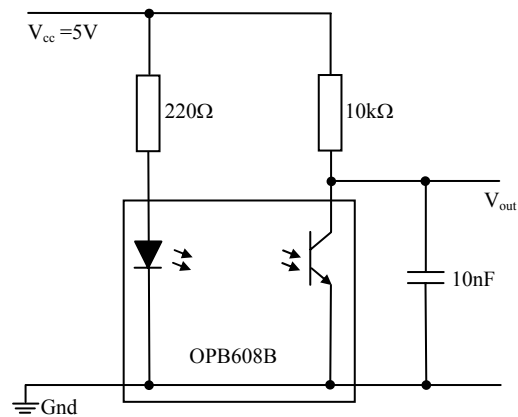
Typical Performance Curves



Optek reserves the right to make changes at any time in order to improve design and to supply the best product possible.
 Optek Technology, Inc. 1215 W. Crosby Road Carrollton, Texas 75006 (972)323-2200 Fax (972)323-2396

Appendix F: Sensor PCB and Circuit

The connection diagram illustrated below shows the actual circuit of the optical sensor. The picture below shows the component side of the PCB.



Appendix G: Dual Motor Drive Data Sheet (extract)

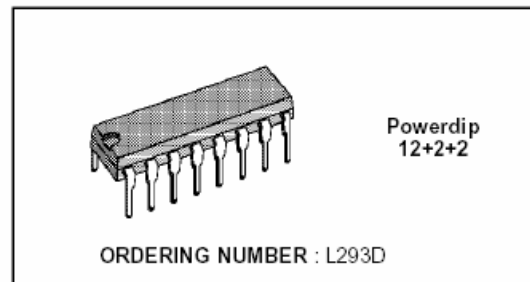


L293D

PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES

PRELIMINARY DATA

- 600mA. OUTPUT CURRENT CAPABILITY PER CHANNEL
- 1.2A PEAK OUTPUT CURRENT (NON REPETITIVE) PER CHANNEL
- ENABLE FACILITY
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5V (HIGH NOISE IMMUNITY)
- INTERNAL CLAMP DIODES



DESCRIPTION

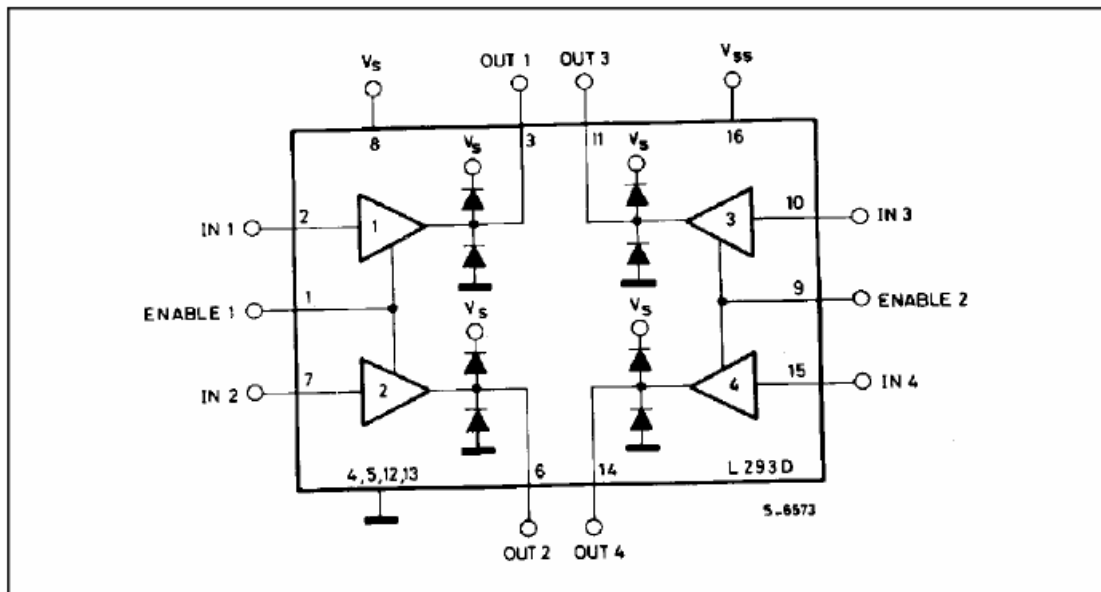
The L293D is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoids, DC and stepping motors) and switching power transistors.

To simplify use as two bridges is pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a low voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 KHz.

The L293D is assembled in a 16 lead plastic package which has 4 center pins connected together and used for heatsinking.

BLOCK DIAGRAM



Appendix H: 8 Channel serial 10 bit ADC Data Sheet

19-0247, Rev. 0, 3/94

MAXIM

Low-Power, 8-Channel, Serial 10-Bit ADC

MAX192

General Description

The MAX192 is a low-cost, 10-bit data-acquisition system that combines an 8-channel multiplexer, high-bandwidth track/hold, and serial interface with high conversion speed and ultra-low power consumption. The device operates with a single +5V supply. The analog inputs are software configurable for single-ended and differential (unipolar/bipolar) operation.

The 4-wire serial interface connects directly to SPI™, QSPI™, and Microwire™ devices, without using external logic. A serial strobe output allows direct connection to TMS320 family digital signal processors. The MAX192 uses either the internal clock or an external serial-interface clock to perform successive approximation A/D conversions. The serial interface can operate beyond 4MHz when the internal clock is used. The MAX192 has an internal 4.096V reference with a drift of ± 30 ppm typical. A reference-buffer amplifier simplifies gain trim and two sub-LSBs reduce quantization errors.

The MAX192 provides a hardwired SHDN pin and two software-selectable power-down modes. Accessing the serial interface automatically powers up the device, and the quick turn-on time allows the MAX192 to be shut down between conversions. By powering down between conversions, supply current can be cut to under 10 μ A at reduced sampling rates.

The MAX192 is available in 20-pin DIP and SO packages, and in a shrink-small-outline package (SSOP) that occupies 30% less area than an 8-pin DIP. The data format provides hardware and software compatibility with the MAX186/MAX188. For anti-aliasing filters, consult the data sheets for the MAX291-MAX297.

Applications

Automotive
Pen-Entry Systems
Consumer Electronics
Portable Data Logging
Robotics
Battery-Powered Instruments, Battery Management
Medical Instruments

See last page for Typical Operating Circuit.

™ SPI and QSPI are trademarks of Motorola Corp.
Microwire is a trademark of National Semiconductor Corp.

Maxim Integrated Products 1

Call toll free 1-800-998-8800 for free samples or literature.

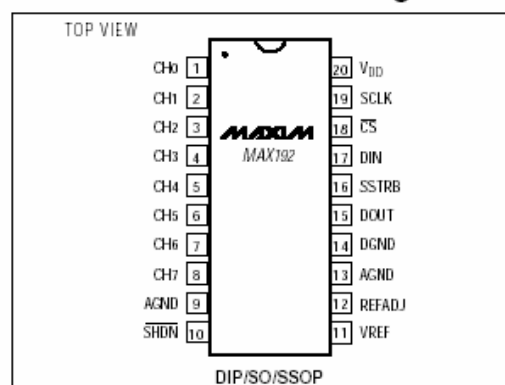
Features

- ◆ 8-Channel Single-Ended or 4-Channel Differential Inputs
- ◆ Single +5V Operation
- ◆ Low Power: 1.5mA (operating)
2 μ A (power-down)
- ◆ Internal Track/Hold, 133kHz Sampling Rate
- ◆ Internal 4.096V Reference
- ◆ 4-Wire Serial Interface is Compatible with SPI, QSPI, Microwire, and TMS320
- ◆ 20-Pin DIP, SO, SSOP Packages
- ◆ Pin-Compatible 12-Bit Upgrade (MAX186/MAX188)

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE	INL(LSBs)
MAX192ACPP	0°C to +70°C	20 Plastic DIP	$\pm 1/2$
MAX192BCPP	0°C to +70°C	20 Plastic DIP	± 1
MAX192ACWP	0°C to +70°C	20 Wide SO	$\pm 1/2$
MAX192BCWP	0°C to +70°C	20 Wide SO	± 1
MAX192ACAP	0°C to +70°C	20 SSOP	$\pm 1/2$
MAX192BCAP	0°C to +70°C	20 SSOP	± 1
MAX192AEPP	-40°C to +85°C	20 Plastic DIP	$\pm 1/2$
MAX192BEPP	-40°C to +85°C	20 Plastic DIP	± 1
MAX192AEWP	-40°C to +85°C	20 Wide SO	$\pm 1/2$
MAX192BEWP	-40°C to +85°C	20 Wide SO	± 1
MAX192AEAP	-40°C to +85°C	20 SSOP	$\pm 1/2$
MAX192BEAP	-40°C to +85°C	20 SSOP	± 1
MAX192AMJP	-55°C to +125°C	20 CERDIP	$\pm 1/2$
MAX192BMJP	-55°C to +125°C	20 CERDIP	± 1

Pin Configuration



Low-Power, 8-Channel, Serial 10-Bit ADC

MAX192

ABSOLUTE MAXIMUM RATINGS

V _{DD} to AGND.....	-0.3V to +6V	SSOP (derate 8.00mW/°C above +70°C)	640mW
AGND to DGND.....	-0.3V to +0.3V	CERDIP (derate 11.11mW/°C above +70°C)	889mW
CH0-CH7 to AGND, DGND	-0.3V to (V _{DD} + 0.3V)	Operating Temperature Ranges	
VREF to AGND	-0.3V to (V _{DD} + 0.3V)	MAX192_C_P	0°C to +70°C
REFADJ to AGND.....	-0.3V to (V _{DD} + 0.3V)	MAX192_E_P	-40°C to +85°C
Digital Inputs to DGND.....	-0.3V to (V _{DD} + 0.3V)	MAX192_MJP	-55°C to +125°C
Digital Outputs to DGND.....	-0.3V to (V _{DD} + 0.3V)	Storage Temperature Range.....	-60°C to +150°C
Continuous Power Dissipation (T _A = +70°C)		Lead Temperature (soldering, 10sec)	+300°C
Plastic DIP (derate 11.11mW/°C above +70°C)	889mW		
SO (derate 10.00mW/°C above +70°C)	800mW		

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

(V_{DD} = 5V ±5%, f_{CLK} = 2.0MHz, external clock (50% duty cycle), 15 clocks/conversion cycle (133ksps), 4.7µF capacitor at VREF pin, T_A = T_{MIN} to T_{MAX}, unless otherwise noted. Typical values are at T_A = +25°C.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
DC ACCURACY (Note 1)						
Resolution			10			Bits
Relative Accuracy (Note 2)		MAX192A			±1/2	LSB
		MAX192B			±1	
Differential Nonlinearity	DNL	No missing codes over temperature			±1	LSB
Offset Error					±2	LSB
Gain Error		External reference, 4.096V			±2	LSB
Gain Temperature Coefficient		External reference, 4.096V		±0.8		ppm/°C
Channel-to-Channel Offset Matching				±0.1		LSB
DYNAMIC SPECIFICATIONS (10kHz sine-wave input, 4.096V_{PP}, 133ksps, 2.0MHz external clock)						
Signal-to-Noise + Distortion Ratio	SINAD			66		dB
Total Harmonic Distortion (up to the 5th harmonic)	THD			-70		dB
Spurious-Free Dynamic Range	SFDR			70		dB
Channel-to-Channel Crosstalk		65kHz, V _{IN} = 4.096V _{PP} (Note 3)		-75		dB
Small-Signal Bandwidth		-3dB rolloff		4.5		MHz
Full-Power Bandwidth				800		kHz
CONVERSION RATE						
Conversion Time (Note 4)	t _{CONV}	Internal clock	5.5		10	µs
		External clock, 2MHz, 12 clocks/conversion	6			
Track/Hold Acquisition Time	t _{AZ}				1.5	µs
Aperture Delay				10		ns
Aperture Jitter				<50		ps
Internal Clock Frequency				1.7		MHz

Low-Power, 8-Channel, Serial 10-Bit ADC

MAX192

ELECTRICAL CHARACTERISTICS (continued)

($V_{DD} = 5V \pm 5\%$, $f_{CLK} = 2.0\text{MHz}$, external clock (50% duty cycle), 15 clocks/conversion cycle (133ksps), 4.7 μF capacitor at VREF pin, $T_A = T_{MIN}$ to T_{MAX} , unless otherwise noted. Typical values are at $T_A = +25^\circ\text{C}$.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS	
External Clock Frequency		External compensation, 4.7 μF	0.1		2.0	MHz	
		Internal compensation (Note 5)	0.1		0.4		
		Used for data transfer only		10			
ANALOG INPUT							
Analog Input Voltage		Common-mode range (any input) (Note 6)	0		V_{DD}	V	
		Single-ended range (unipolar only)	0		V_{REF}		
		Differential range	Unipolar	0			V_{REF}
			Bipolar	$-\frac{V_{REF}}{2}$			$+\frac{V_{REF}}{2}$
Multiplexer Leakage Current		On/off leakage current, $V_{IN} = 0V, 5V$		± 0.01	± 1	μA	
Input Capacitance		(Note 5)		16		pF	
INTERNAL REFERENCE (reference buffer enabled)							
VREF Output Voltage		$T_A = +25^\circ\text{C}$	4.076	4.096	4.116	V	
VREF Short-Circuit Current					30	mA	
VREF Tempco				± 30		ppm/ $^\circ\text{C}$	
Load Regulation (Note 7)		0mA to 0.5mA output load		2.5		mV	
Capacitive Bypass at VREF		Internal compensation	0			μF	
		External compensation	4.7				
Capacitive Bypass at REFADJ		Internal compensation	0.01			μF	
		External compensation	0.01				
REFADJ Adjustment Range				± 1.5		%	
EXTERNAL REFERENCE AT VREF (buffer disabled, $V_{REF} = 4.096\text{V}$)							
Input Voltage Range			2.5		$V_{DD} + 50\text{mV}$	V	
Input Current				200	350	μA	
Input Resistance			12	20		k Ω	
Shutdown VREF Input Current				1.5	10	μA	
Buffer Disable Threshold REFADJ			$V_{DD} - 50\text{mV}$			V	
EXTERNAL REFERENCE AT REFADJ							
Capacitive Bypass at VREF		Internal compensation mode	0			μF	
		External compensation mode	4.7				
Reference-Buffer Gain				1.678		V/V	
REFADJ Input Current					± 50	μA	

Low-Power, 8-Channel, Serial 10-Bit ADC

MAX192

ELECTRICAL CHARACTERISTICS (continued)

($V_{DD} = 5V \pm 5\%$, $f_{CLK} = 2.0\text{MHz}$, external clock (50% duty cycle), 15 clocks/conversion cycle (133ksps), 4.7 μF capacitor at VREF pin, $T_A = T_{MIN}$ to T_{MAX} , unless otherwise noted. Typical values are at $T_A = +25^\circ\text{C}$.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
DIGITAL INPUTS (DIN, SCLK, CS, SHDN)						
DIN, SCLK, CS Input High Voltage	V_{INH}		2.4			V
DIN, SCLK, CS Input Low Voltage	V_{INL}				0.8	V
DIN, SCLK, CS Input Hysteresis	V_{HYST}			0.15		V
DIN, SCLK, CS Input Leakage	I_{IN}	$V_{IN} = 0V$ or V_{DD}			± 1	μA
DIN, SCLK, CS Input Capacitance	C_{IN}	(Note 5)			15	pF
SHDN Input High Voltage	V_{INH}		$V_{DD} - 0.5$			V
SHDN Input Low Voltage	V_{INL}				0.5	V
SHDN Input Current, High	I_{INH}	SHDN = V_{DD}			4.0	μA
SHDN Input Current, Low	I_{INL}	SHDN = 0V	-4.0			μA
SHDN Input Mid Voltage	V_{IM}		1.5	$V_{DD} - 1.5$		V
SHDN Voltage, Floating	V_{FLT}	SHDN = open		2.75		V
SHDN Max Allowed Leakage, Mid Input		SHDN = open	-100		100	nA
DIGITAL OUTPUTS (DOUT, SSTRB)						
Output Voltage Low	V_{OL}	$I_{SINK} = 5\text{mA}$			0.4	V
		$I_{SINK} = 16\text{mA}$		0.3		
Output Voltage High	V_{OH}	$I_{SOURCE} = 1\text{mA}$	4			V
Three-State Leakage Current	I_L	CS = 5V			± 10	μA
Three-State Leakage Capacitance	C_{OUT}	CS = 5V (Note 5)			15	pF
POWER REQUIREMENTS						
Positive Supply Voltage	V_{DD}			$5 \pm 5\%$		V
Positive Supply Current	I_{DD}	Operating mode		1.5	2.5	μA
		Fast power-down		30	70	
		Full power-down		2	10	
Positive Supply Rejection (Note 8)	PSR	$V_{DD} = 5V \pm 5\%$; external reference, 4.096V; full-scale input		± 0.06	± 0.5	mV

Note 1: Tested at $V_{DD} = 5.0V$; single-ended, unipolar.

Note 2: Relative accuracy is the deviation of the analog value at any code from its theoretical value after the full-scale range has been calibrated.

Note 3: Grounded on-channel; sine wave applied to all off channels.

Note 4: Conversion time defined as the number of clock cycles times the clock period; clock has 50% duty cycle.

Note 5: Guaranteed by design. Not subject to production testing.

Note 6: The common-mode range for the analog inputs is from AGND to V_{DD} .

Note 7: External load should not change during conversion for specified accuracy.

Note 8: Measured at $V_{SUPPLY} +5\%$ and $V_{SUPPLY} -5\%$ only.

Appendix I: Added C++ functions

In this section, the most important functions added and their usage are listed. When necessary, an error management is also implemented in the function and the error is displayed on the LCD of the EyeBot.

LCD Output

```
void LCDCircle(int x, int y, int rad)
```

Input: x, y position of the centre of circle
rad is the radius of the desired circle

Output: NONE

Semantics: Prints a circle on the LCD with the centre at x,y and the radius of rad (This function uses the MySin an the MyCos functions, which needed the lookup-table implemented in file RSmath.c)

```
void LCDCross(int x, int y, int val)
```

Input: x, y position of the centre of the cross
val is the value of the pixel operation code

0= clear pixel
1= set pixel
2= invert pixel

Output: NONE

Semantics: Prints a cross on the LCD with the centre at x,y and pixel code of val

```
void LCDDeg(int x, int y, int val)
```

Input: x, y position of the centre of the degree sign
val is the value of the pixel operation code

0= clear pixel
1= set pixel
2= invert pixel

Output: NONE

Semantics: Prints a degree sign on the LCD with the centre at x,y and pixel code of val

Math functions

float MySin(int angle)

Input: Angle that is to be calculated the sinus from

Output: Sinus of the input

Semantics: Fast calculation of a sinus using the lookup-table implemented in the file RSmath.c (Accuracy of 2 values per degree!)

float MySin(int angle)

Input: Angle that is to be calculated the cosinus from

Output: cosinus of the input

Semantics: Fast calculation of a cosinus using the lookup-table implemented in the file RSmath.c (Accuracy of 2 values per degree!)

int Limit(int x, int uplimit, int dolimit)

Input: x, value to be limited
uplimit, value of the upper limit
dolimit, value of the lower limit

Output: limited value

Semantics: Builds the limit of value, if this value exceeds the limits

int Round(double x)

Input: x, value to be rounded

Output: rounded value

Semantics: Calculates a rounded int out of a float

int AnalogSensor (int x)

Input: x, value of the channel to be read out

Output: actual value of the analog input channel

Semantics: Returns the value of the desired analog input channel waiting the need time for the ADC to ensure a reliable value

float Sen2Angle (float value)

Input: value to be transformed into an angle

Output: angle of the inputted analog value

Semantics: Returns the angle of the analog value inputted

Initialization

```
void QuIni()  
  Input:    NONE  
  Output:   NONE  
  Semantics: Initializes the Quad-encoders of the three used motors
```

```
void MoIni()  
  Input:    NONE  
  Output:   NONE  
  Semantics: Initializes the three used motors
```

```
void MOQuRelease()  
  Input:    NONE  
  Output:   NONE  
  Semantics: Releases the three used encoders and the motors
```

Calibration

```
void CaliFeet()  
  Input:    NONE  
  Output:   NONE  
  Semantics: Calibrates the legs using the optical sensors to reset the encoders at  
             a defined position
```

```
void CaliIncl()  
  Input:    NONE  
  Output:   NONE  
  Semantics: Calibrates the inclinometer (Until the sensor is not attached firmly  
             to the robot, this calibration is inevitable)
```

```
int CaliWeight()  
  Input:    NONE  
  Output:   Maximal value of the possible movement in both directions  
  Semantics: Calibrates the moving mass system resetting the encoder of the  
             motor at the centre of the system and returns the value of the  
             maximal movement in both directions
```

Control

void FeetControl()

Input: NONE

Output: NONE

Semantics: Starts the controller for the movement of both legs with the parameters specified in the function `init`

void HipBalance (int p_f_l, int p_f_r)

Input: position of the left and the right leg

Output: NONE

Semantics: Starts the controller for the movement of both legs with the parameters specified in the function `init` but with the speed zero. The inputted position will be tried to hold.

void WeightMove(int pos, int max, int speed)

Input: position to be moved the mass
maximal possible position
desired speed at which to be moved the mass

Output: NONE

Semantics: Moves the moving mass to desired position at a desired speed.

void WeightControl (double angle_lr, int max)

Input: value of the actual angle
maximal possible position

Output: NONE

Semantics: Starts the controller for the moving mass system for static control.

Declaration

I hereby declare that this submission is my own work and that I only used the referenced aids.

Perth,

Antonio Pickel