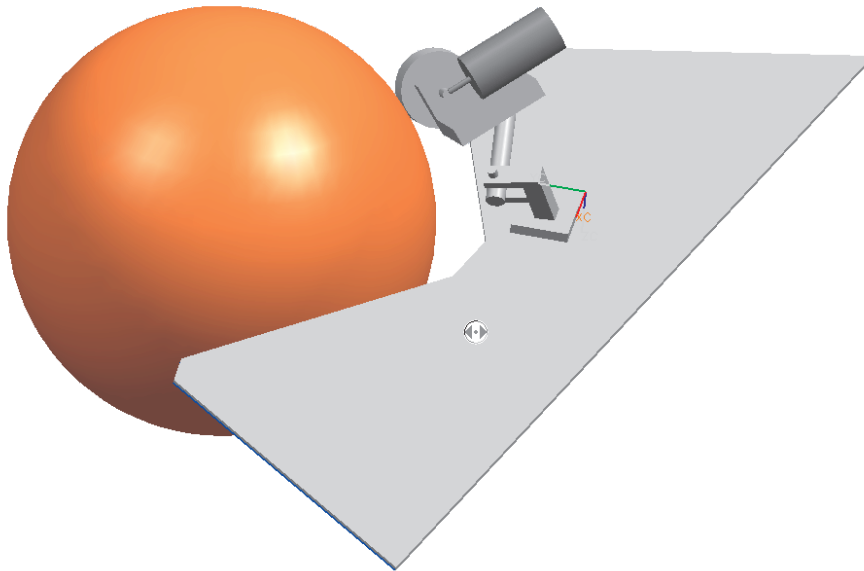


# Design of a Ball Handling Mechanism for Robocup

Bachelor End Project

Rob Hoogendijk

DCT 2007.018



Supervised by:  
dr. ir. M. J. G. van de Molengraft  
ir. W. H. T. M. Aangenent  
ir. R. J. E. Merry

Bachelor End Project, departement Mechanical Engineering,  
Technische Universiteit Eindhoven



## **Abstract**

This report describes the design, realization and testing of a ball handling mechanism for Tech United, the Robocup team of the Technische Universiteit Eindhoven. The goal of this project is to enable the robot to receive the ball and to dribble with the ball. Various principles for stopping the ball and dribbling with the ball are evaluated and the concept that uses two wheels attached to levers is chosen. The levers are used to exert a force on the ball to get good grip on the ball. The ball can be caught by clamping it between these two levers. A prototype of the designed ball handling concept was realized. After mechanical and electrical implementation on a TURTLE robot a PI feedback controller has been designed and tested. The performance of the prototype is very promising. The ball stays in contact with the robot when driving forwards, backwards and when braking.



# **preface**

This report is written as the closure of the Bachelor End Project for my study Mechanical Engineering at the Technische Universiteit Eindhoven. The reserved time for this project is 224 hours, which is equal to 8 ECTS. I have worked on this project from September until Januari 2006 and have invested more time because the project was very interesting and fun.

I acquired a lot of knowledge from this project about a very broad range of topics because of its multidisciplinary character. Beginning with this report itself, this is my first report written with LaTeX and it has been a very pleasant and instructive acquaintance. Furthermore, I learnt a lot about design processes, its relation to realization possibilities in mechanics and electronics and I have refreshed my knowledge about system identification and controller design.

Because of the multidisciplinary character of the project, I could not have done this alone. The whole Tech United team supported me with advice and help and I offer my gratitude to the whole team. Special thanks to Wouter Aangenent who was always available for my questions, and helped me with the software of the robot. Also thanks to Roel Merry for feedback and motivation and to Rob van den Berg and Harrie van de Loo for assistance on the mechanical and electrical realization.

I enjoyed working on this project because of the possibility to see the realization and performance of a design. I am pleased to see that others will continue with my concept because of its promising results and high potential.



# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
<b>2</b>	<b>Design Criteria</b>	<b>9</b>
2.1	Rules and Regulations . . . . .	9
2.2	Available Space . . . . .	10
<b>3</b>	<b>Concepts</b>	<b>11</b>
3.1	Stopping the Ball . . . . .	11
3.2	Dribbling . . . . .	12
3.3	Concept Selection . . . . .	13
3.3.1	Stopping . . . . .	13
3.3.2	Dribbling . . . . .	14
3.3.3	Stopping and Dribbling Combined . . . . .	14
<b>4</b>	<b>Prototype Realization</b>	<b>17</b>
4.1	Mechanical Realization . . . . .	17
4.2	Electronic Realization . . . . .	18
4.2.1	Power Estimation . . . . .	18
4.2.2	Velocity Calculation . . . . .	18
4.2.3	Layout of the TURTLE robot . . . . .	21
<b>5</b>	<b>Identification and Control</b>	<b>23</b>
5.1	System Identification . . . . .	23
5.2	Controller Design . . . . .	25
5.3	Reference Velocity Generation . . . . .	26
5.4	Implementation . . . . .	26
<b>6</b>	<b>Performance Evaluation</b>	<b>29</b>
<b>7</b>	<b>Conclusion and Recommendations</b>	<b>33</b>
7.1	Conclusion . . . . .	33
7.2	Recommendations . . . . .	33
<b>A</b>	<b>Datasheets DC Motor and Tachometer</b>	<b>37</b>
<b>B</b>	<b>Source Code Controller</b>	<b>41</b>





# Chapter 1

## Introduction

RoboCup is an international project that promotes AI, robotics, and related fields. It is an attempt to foster AI and intelligent robotics research by providing a standard problem where a wide range of technologies can be integrated and examined. RoboCup chose to use the soccer game as a central topic of research, aiming at innovations to be applied for socially significant problems and industries. The ultimate goal of the RoboCup project is:

By 2050, develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer.[1]

Robocup consists of various leagues. There are separate soccer leagues for various wheeled, legged and humanoid robots, but there are also simulation and rescue leagues. The Robocup team of the Technische Universiteit Eindhoven, Tech United, competes in the Middle Size League, in which wheeled robots autonomously play soccer.

This report describes the design, realization and testing of a ball dribbling mechanism for the TURTLE (Tech United Robot Limited Edition) robots. Various efforts have been made to build a mechanism to control the ball, but because of changing rules and room for improvement of the current design, a new project on this topic was started.

The goal of this project is to enable the robot to receive the ball and dribble forwards and backwards with the ball. Turning and driving sideways are not the aim of this project because of lack of time. However, the design can be expanded to cope with turning and driving sideways. This will be topic of future research because this skill is crucial for the succes of the TURTLE robots in real match conditions.

The outline of the report is as follows. First, the design criteria are presented in Chapter 2. Chapter 3 discusses the basic concepts of stopping the ball and dribbling with the ball and the selection of the best concept. This is followed by a description of the realization of the design in Chapter 4 and the system identification and controller design in Chapter 5. The testing of the design is discussed in Chapter 6. Finally, the results of these tests and the design itself are evaluated in Chapter 7.



## Chapter 2

# Design Criteria

This chapter describes the demands that are placed on the design. First, the Robocup Rules and Regulations and its implications on the design are discussed. Next, the available space in the robot for the design is described.

### 2.1 Rules and Regulations

The Rules and Regulations [1] of the soccer game are as close as possible to the FIFA rules. However, there are special rules that describe what is allowed in ball handling. The part of the Rules and Regulations that is the relevant for the design is:

- During a game, the ball must not enter the convex hull of a robot by more than a third of its diameter except when the robot is stopping the ball. The ball must not enter the convex hull of a robot by more than half of its diameter if the robot is stopping the ball. This case only applies to instantaneous contact between robot and ball lasting no longer than one second. In any case it must be possible for another robot to take possession of the ball.
- The robot may exert a force onto the ball only by direct physical contact between robot and ball. Forces exerted onto the ball that hinder the ball from rotating in its natural direction of rotation are allowed for no more than four seconds and a maximal distance of movement of one meter. Exerting this kind of forces repeatedly is allowed only after a waiting time of at least four seconds. Natural direction of rotation means that the ball is rotating in the direction of its movement.
- Violating any of the above rules is considered ball holding.

The first item of the rules limits the enclosure of the ball. Fig. 2.1 visualizes how this affects the allowed position of the ball relative to the robot. The dotted line indicates the boundary of the “convex hull” that is mentioned in the Rules and Regulations. The ball is allowed to enter this hull for one third of its diameter when the robot is dribbling. If the robot stops the ball it is allowed to enclose half the diameter of the ball, but only for one second.

The second item of the rules describes how to apply force to the ball. This will be of importance in the discussion of the various principles that are discussed in Chapter 3. If ball holding is observed by the referee, an indirect free kick is awarded to the opposing team. Therefore compliance to this rules is essential for a successful robot soccer team.

Another rule is that the maximum weight that is allowed for a robot is limited to 40 kg. Because the frame, batteries and electronics already have a substantial weight, the design of the ball handling mechanism should be as light as possible.

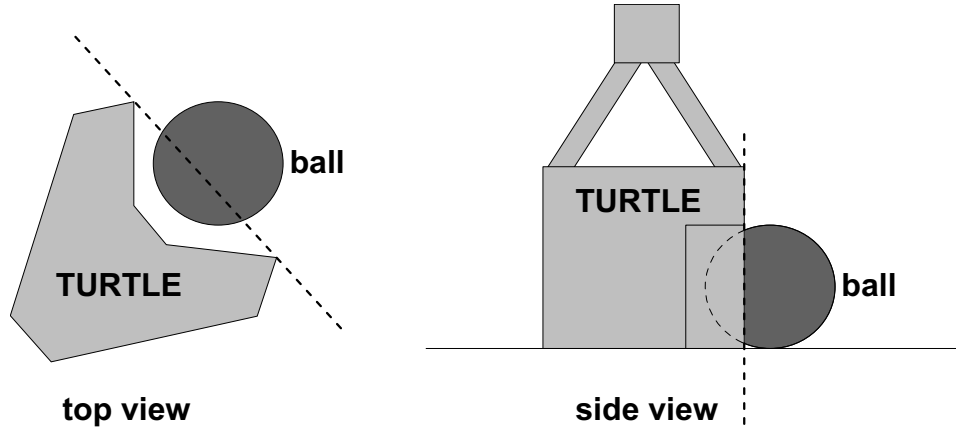


Figure 2.1: Top and side view of the turtle robot and the ball. The dotted lines indicate the boundaries of the convex hull of the TURTLE robot. The ball must not enter this area for more than one third when dribbling with the ball or one half when the robot is stopping the ball.

## 2.2 Available Space

A TURTLE robot consists of three main layers as is depicted in Fig. 2.2A. The first layer is filled with the three motors (M) that drive the robot, and the batteries. The second layer contains the shooting mechanism (S). Two TUEDACs (TU/e Data Acquisition and Control System) units (T) occupy most space in the third layer. These units form the connection between the electronics of the TURTLE robot and the laptop (L) that is located on top of the third layer. The space that is available for the dribbling mechanism (D) is located at the two sides of the shooting mechanism (S). An indication of the dimensions of this area is given in Fig. 2.2B.

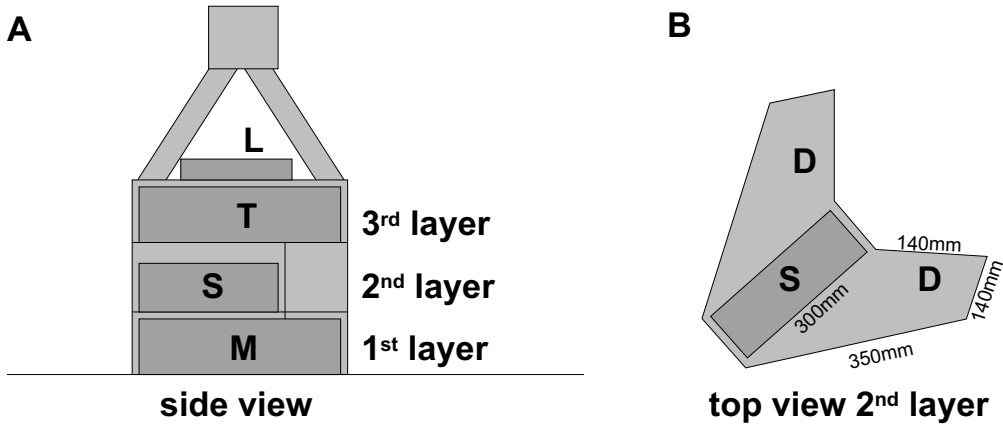


Figure 2.2: A TURTLE robot consists of three layers. The first layer is filled with the motors and batteries (M). The second layer contains the shooting mechanism (S). Two TUEDACs (T) units occupy most space in the third layer. These units are the connection between the electronics and the laptop (L) that is placed on top of the robot. The available space for the dribbling mechanism (D) is situated in the second layer at two sides of the shooting mechanism (S).

## Chapter 3

# Concepts

Some basic principles that can be used to stop and dribble with a soccer ball are discussed in this chapter. The concepts were gathered from other Robocup leagues, from Tech United team members and from own ideas. This chapter gives an overview of these concepts and the process of choosing a concept. First, the stopping and dribbling concepts are discussed, followed by a comparison of the various concepts.

### 3.1 Stopping the Ball

As a first idea, the way a human soccer player stops a soccer ball when it is passed to him is analyzed. A soccer player places his foot against the ball with a force and speed such that the ball comes to a halt. Fig. 3.1A shows how this could be realized with the robot. The robot drives against the ball with a speed  $v$  and applies a force  $F$  to the ball, that approaches the robot with a speed  $v_{ball}$ . This applied force and the speed of the robot will probably have to vary during contact with the ball to achieve a zero end velocity of the ball.

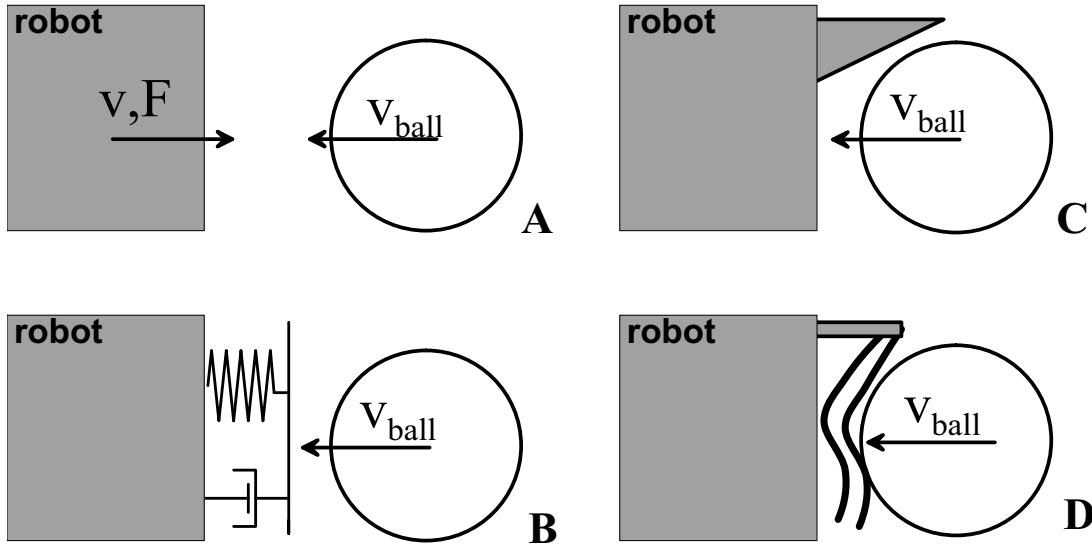


Figure 3.1: Various concepts to stop a soccer ball approaching the robot with velocity  $v_{ball}$ . A: controlled collision with the robot that drives against the ball with velocity  $v$  and applies a force  $F$ , B: spring damper system, C: clamping mechanism, D: “curtain”

The second idea is to use some kind of spring-damper system that dissipates the energy of the ball. The damping coefficient of this system should be adapted to the mass of the ball in such a

way that the velocity of the ball is zero after interaction with this system. This could be realized either by a spring-damper system as in Fig. 3.1B, or by a suitable foam or gel. The Cornell university team reported great improvement of catching capabilities when they added a spring damper system to the dribbling system of their Small Size League robot[2].

Another way of stopping the ball is to clamp it. This could either be done by a passive element such as an opening that is smaller than the ball such as the wedge shaped configuration shown in Fig. 3.1C, or by an active clamping mechanism.

The last principle investigated is based on friction. To stop the ball, it is brought in contact with a large surface. This concept, visualized in Fig. 3.1D, has already been tested by Tech United. The “curtain”, as it is called, is made of rubber or out of small metal rings like a Medieval chain mail. When the ball rolls into the curtain, it assumes the shape of the ball generating friction with the surface and in the case of the chain mail curtain, also within the curtain itself.

## 3.2 Dribbling

The currently used dribbling mechanism, shown in Fig. 3.2A, simply consists of a small fork, shaped to fit around the ball. With this device, the ball’s momentum carries it away from the robot when decelerating. Also when driving sideways or in reverse, the robot moves away from the ball. Therefore, a new mechanism that always keeps the ball against the robot is desirable.

A human soccer player controls the ball by giving it small kicks in the right direction. For a robot this would require a very fast detection and actuation system. A device that stays in contact with the ball constantly would be a simpler solution. This is exactly what the following concepts do.

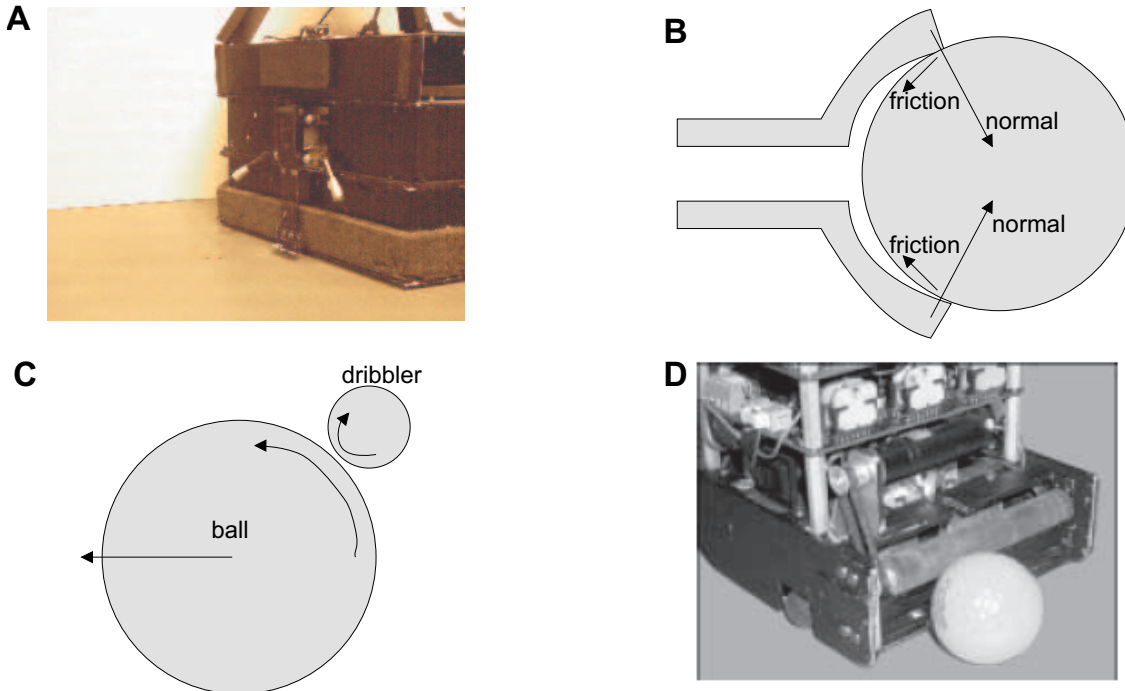


Figure 3.2: Various dribbling concepts. A: current dribbling mechanism, B: Force feedback dribbler, C: dribbling with wheels, [2], D: bar dribbler, [6].

As a first concept for dribbling, force-feedback is considered, see Fig. 3.2B. The ball is grabbed between two “fingers” which apply a force such that the ball stays in contact with the robot, but at the same time allow it to roll in its natural direction. This is required by the rules, as mentioned in Chapter 2. This system controls the friction force applied to the ball by changing the normal force on the ball.

A second concept is the use of one or more rotating elements that keep contact with the ball. Fig. 3.2C shows such a principle. The rotational velocity of the dribbler is related to velocity of the robot. The positioning of the rotational elements will probably affect their performance. Rotating elements placed on top of the ball have good control for braking and reverse driving, while wheels on the side of the ball can control the lateral position of the ball better.

In the Small Sized League, bar dribblers such as the one shown in Fig. 3.2D are commonly used [3],[4],[5]. A rotating bar, placed horizontally on the robot, controls the ball by spinning it against its natural direction. Vertical bars on the side can prevent the ball from rolling sideways. Although counter rotation mechanisms are not allowed in the Middle Size League, these bars could also be rotated in the direction of the movement of the robot, making the ball roll in its natural direction. In stead of rotating bars, wheels could also be used for this purpose.

### 3.3 Concept Selection

The various concepts are evaluated on basis of compliance to the Robocup rules, potential and complexity of realization. Each of these three criteria is equally important. Breaking the rules can lead to free kicks for the opposing team or even to disqualification. The performance of the dribbling mechanism has to be good in order to win in Robocup matches. The complexity of realization is important because the reliability of a system often decreases with increasing complexity. To quantify the evaluation, grades from 1(very poor) to 5(very good)are given to each concept, based on an estimation of agreement to these criteria.

#### 3.3.1 Stopping

Table 3.1 shows the quantification of the evaluation of the stopping concepts. The explanation of these grades is as follows.

Stopping the ball by means of controlled collision will probably be hard to realize because the velocity of the ball must be estimated precisely and the actuation of the robot will be complex. This method might not be effective because the ball will probably not stop completely. A spring damper system has proven to work in the Small Sized League[2], but it will result in a complexer system that is more sensitive to failures. A foam or gel damper is simple to realize, but finding a foam or gel with the right properties is expected to be difficult. A clamping mechanism that is only allowed to grab one third of the ball will probably not get a good grip on the ball, therefore this mechanism could be less effective. Also, it prevents the natural direction of rotation which could break the rules. However, it will be easy to manufacture and not complex and therefore robust. First test with a prototype of the curtain concept showed that it can stop the ball effectively. A possible disadvantage could be that the curtain can get stuck to other robots because it hangs loosely in front of the robot. An other possible disadvantage could be that it could interfere with the shooting and dribbling mechanism.

The damping, clamping or curtain concepts are probably the most promissing. Build-in space, weight, and especially integration with the dribbling mechanism will determine which design is best for this situation.

Concept	Rules	Potential	Realization	Total
collision	5	1	1	7
damping:spring damper	5	4	3	12
damping:foam	5	3	3	11
clamping	4	3	5	12
curtain	5	4	3	12

Table 3.1: Evaluation of stopping concepts.

### 3.3.2 Dribbling

Quantification of the evaluation of the dribbling concepts is given in Table 3.2. The reasoning for these grades is as follows.

The current design was very simple to realize, but it can not deal with driving sideways or in reverse. Braking and turning is only possible around the center of the ball. As mentioned in Section 3.2, trying to dribble like a real soccer player is complex. The position and speed of the ball have to be measured. The actuation has to be fast, precise and multidirectional in order to keep the ball near the robot. The force feedback principle has to let the ball slip in its “fingers” in order to allow it to roll naturally. This will probably cause the ball to slowly roll away when driving in reverse or sideways. It requires a force feedback actuation system that makes realization harder than a passive device. Rotating dribblers can possibly keep contact with the ball when driving sideways or in reverse, rotating and when braking. Realization of a rotating dribbler will also be more complex, because it involves an electric drive and a mechanism to exert force on the ball.

A rotating dribbler is the best choice according to this evaluation.

Concept	Rules	Potential	Realization	Total
passive fork	5	1	5	11
repeated kicks	5	3	1	9
force feedback	5	3	3	11
rotating dribbler	5	5	3	13

Table 3.2: Evaluation of dribbling concepts.

### 3.3.3 Stopping and Dribbling Combined

Based on the evaluation of the dribbling concept, we have chosen to use a rotating dribbler. A design that uses two wheels on two sides of the ball is selected. Fig. 3.3 shows a schematic representation of this concept. This configuration stabilizes the lateral movement of the ball and enables good control when driving sideways or turning. The wheels are positioned on the upper half of the ball, enabling them to keep traction on the ball when driving in reverse. This design also does not interfere with the shooting mechanism, which is located between the two wheels. The wheels are mounted on levers which rotate about a pivot point to be able to control the force exerted on the ball. The levers move towards the ball due to their own weight, exerting a force to get good traction on the ball, see Fig. 3.4B. Springs or rubber bands can be added to increase or decrease this force if this is necessary.

A stopping mechanism has to be included in this design as well. A clamping mechanism is the easiest to realize, because of the presence of the two levers. The two levers are positioned in such a way that the opening between the wheels is smaller than the ball. When the robot catches the ball, the wheels grab the ball and push the levers outwards until the ball is pushed against the robot, see Fig. 3.4A. Thus, the wheel actually improve the clamping concept.



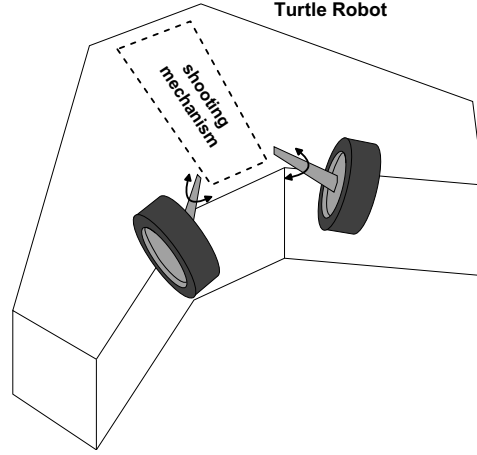


Figure 3.3: Schematic representation of the Turtle Robot and the concept for the dribbling mechanism. Two wheels are placed on levers that can rotate in the direction indicated by the arrows. This concept does not interfere with the shooting mechanism that is indicated by the dotted rectangle.

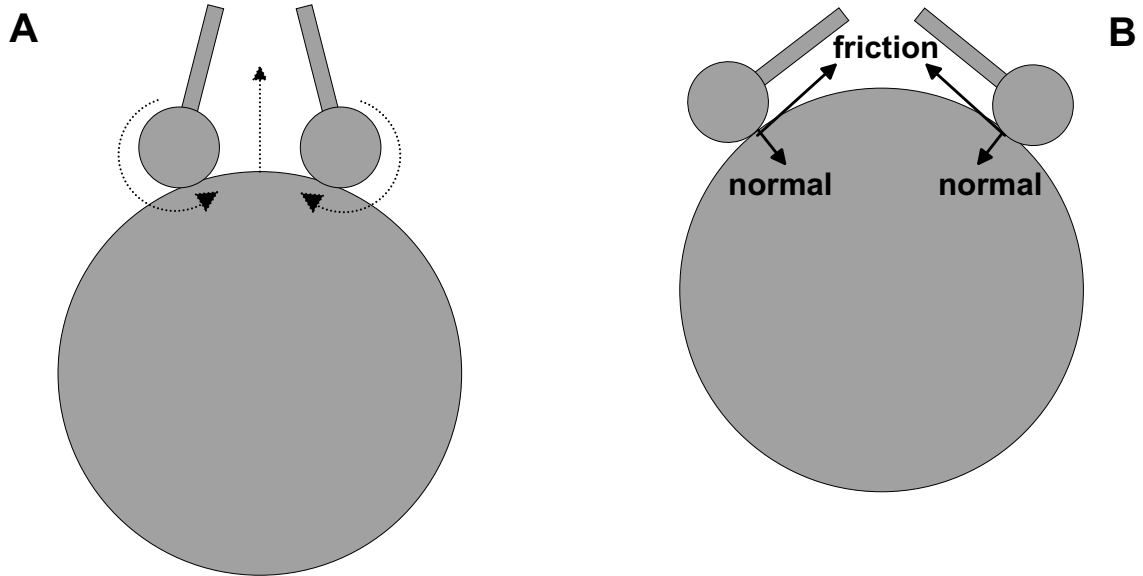


Figure 3.4: Catching the ball (A) and forces on the ball (B), top view. A: To catch the ball, the wheels grab the ball and pull it towards the robot, moving the levers outwards. B: The weight of the mechanism exerts a normal force that causes a friction force that pulls the ball towards the robot.



## Chapter 4

# Prototype Realization

### 4.1 Mechanical Realization

A simple prototype is constructed out of metal profiles. Fig. 4.1 shows a photograph of one half of the mechanism. The length of the lever, the angle of the lever, and the angle of the wheel to the ball can be adjusted. These adjustment possibilities are used to realize a setup that encloses the ball for one third of its diameter, the maximum allowed by the rules. The soft rubber wheel (A),

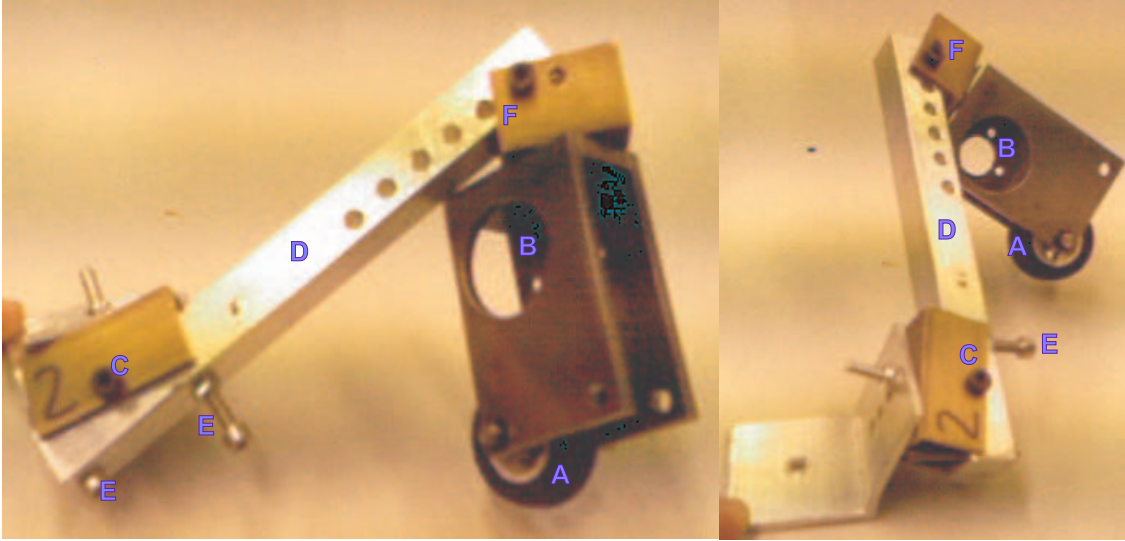


Figure 4.1: Photograph of one half of the mechanics of the prototype viewed from two sides. A: wheel, B: motor bracket, C: pivot point, D: lever, E: adjustment screws, F: adjustable connection.

taken from a Lego Mindstorms kit, provides good grip on the ball. This wheel is mounted in the motor bracket (B) and connected to the motor via pulleys and a toothed belt(not visible). The pivot point (C) is simply realized as a bolt with a lock nut. It enables the lever (D) to rotate freely because the rotational motion is not actuated. Two adjustment screws (E) are used to facilitate easy adjustment of the stroke limits of the lever. The connection (F) of the motor bracket to the lever is adjustable. The bracket can be rotated to change the contact angle between the wheel and the ball and it can be mounted at different positions on the lever.

## 4.2 Electronic Realization

### 4.2.1 Power Estimation

DC motors are used to drive the wheels. An estimation of the power required is used to select the motor specifications. The ball has a mass  $m_b$  of 430 g and a radius  $r_b$  of 110 mm. The system is designed for a velocity  $v$  of the robot of 3 m/s and an acceleration rate  $a$  of 10 m/s<sup>2</sup>. The power  $P$  is equal to the product of torque  $T$  and the angular velocity  $\omega_b = v/r_b$  thus,

$$P = T\omega_b. \quad (4.1)$$

The torque can be calculated from Newtonian mechanics for rotational motion, as

$$T = I\alpha_b, \quad (4.2)$$

where  $\alpha_b = a/r_b$  is the angular acceleration of the ball and  $I$  is the inertia of the ball. The inertia of the ball is estimated from the standard formula for the inertia of a thin hollow sphere with mass  $m_b$  and radius  $r_b$  that is equal to

$$I = \frac{2}{3}m_br_b^2. \quad (4.3)$$

Substituting equation 4.2 and 4.3 in equation 4.1 and rewriting in known variables yields

$$P = \frac{2}{3}mva \approx 8.6 \text{ Watt}. \quad (4.4)$$

Two 10 Watt motors from Maxon Motors were used to have a safety margin, ensuring sufficient power. The datasheet of these motors can be found in appendix A.

### 4.2.2 Velocity Calculation

Fig. 4.2 illustrates that the velocity of the ball is related to the velocity of the robot by

$$v_b = v_x\vec{e}_1 + (v_y + d\omega_r)\vec{e}_2, \quad (4.5)$$

$$\phi = \arctan\left(\frac{v_y + d\omega_r}{v_x}\right), \quad (4.6)$$

where  $v_b$  and  $\phi$  are the velocity of the ball and its angle relative to forward motion.  $v_x$  and  $v_y$  are the velocities of the robot in x and y direction,  $\omega_r$  is the angular velocity of the robot and  $d$  is the distance between the center of the robot and the center of the ball. At first sight one might expect that the velocity of the ball should equal the velocity of the robot. However, a rotation of the robot contributes to the velocity at the position of the ball.

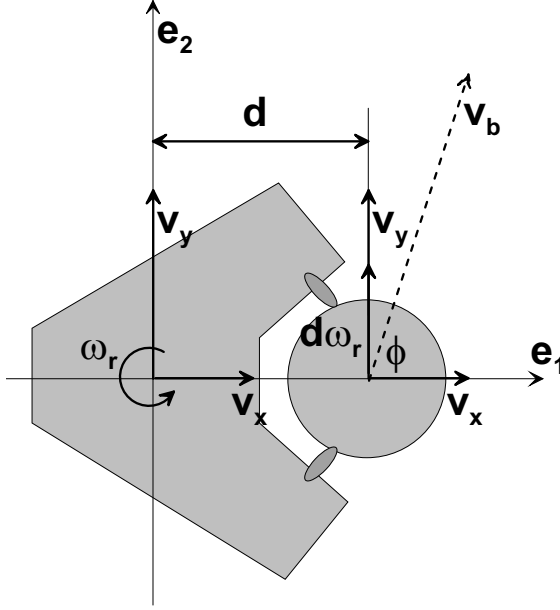


Figure 4.2: Relation between the velocity of the robot and the velocity of the ball.  $v_x$ ,  $v_y$  and  $\omega_r$  denote the x, y and angular components of the velocity of the robot. The distance between the center of the ball and the center of the robot is called  $d$ . The dotted line indicates the resulting velocity of the ball  $v_b$  at an angle  $\phi$ .

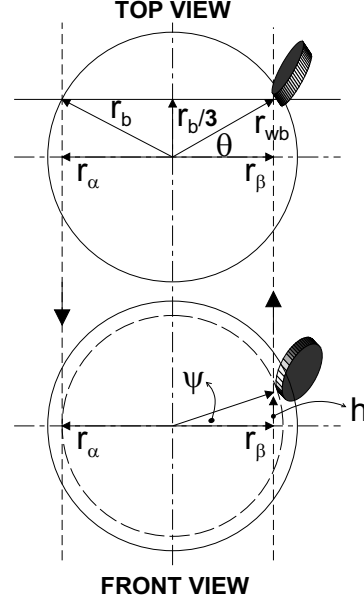


Figure 4.3: The angle  $\theta$ , radius  $r_{wb}$  and height  $h$  can be calculated from the angle  $\psi$  with the help of radii  $r_\alpha$  and  $r_\beta$ . The distance between the wheels is chosen such that the ball enters the convex hull of the robot for one third of its diameter.

The positioning of the wheels determines the relation between the velocity of the ball and the angular velocity of the wheels. The wheels are placed on the upper half of the ball and at a distance relative to each other such that the ball enters the convex hull of the robot for one third of its diameter, which is the maximum allowed by the rules. This is illustrated by Fig. 4.3. The radius  $r_{wb}$ , its angle relative to the lateral direction of the robot  $\theta$ , and the height  $h$  relative to the center of the ball need to be known to calculate the angular velocities of the wheels. To calculate these first the auxiliary radius  $r_\alpha$  is calculated by looking at left side of the top view. This gives

$$r_\alpha = \sqrt{r_b^2 - \left(\frac{1}{3}r_b\right)^2} = \sqrt{\frac{8}{9}}r_b, \quad (4.7)$$

where  $r_b$  is the radius of the ball. From the front view it can be observed that

$$r_\beta = r_\alpha \cos(\psi), \quad (4.8)$$

where  $\psi$  is defined as the angle to the horizontal direction. This angle determines at what height on the upper half of the ball the wheels are placed.  $r_\beta$  is also an auxiliary radius. The height at which the wheels touch the ball, relative to the center of the ball is equal to

$$h = r_\alpha \sin(\psi), \quad (4.9)$$

From the right side of the top view it is clear that

$$r_{wb} = \sqrt{r_\beta^2 + \left(\frac{1}{3}r_b\right)^2} \quad (4.10)$$

and

$$\theta = \arctan\left(\frac{\frac{1}{3}r_b}{r_\beta}\right). \quad (4.11)$$

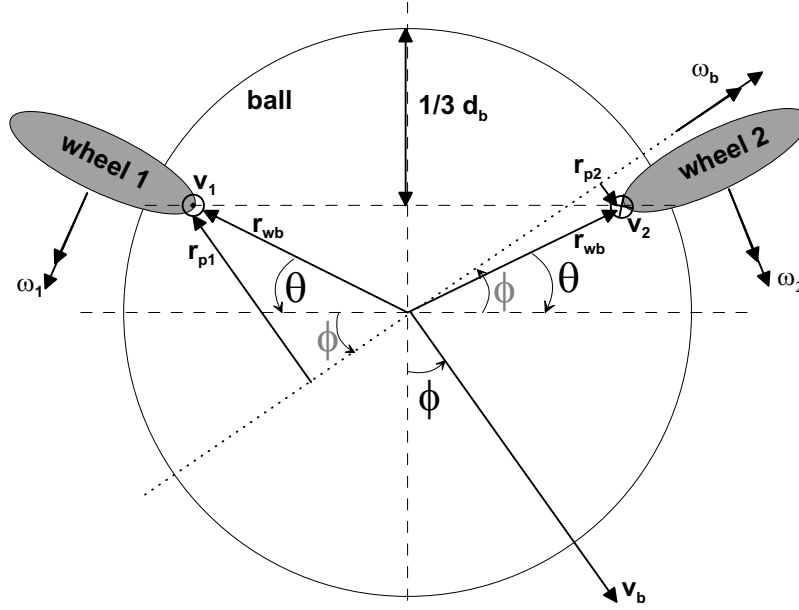


Figure 4.4: Schematic representation of the two wheels and the ball, top view.  $r_{p1}$  and  $r_{p2}$  indicate the projection of the radii at which the first and second wheel rotate on the surface of the ball. The ball has a radius  $r_b$  and diameter  $d_b$ .  $r_{wb}$  is an auxiliary radius for calculations. The angle of the wheels to the ball is indicated with  $\theta$ , while  $\phi$  denotes the angle of the direction of movement of the robot. The velocities at the surface of the wheels are indicated by  $v_1$  (direction=upwards) and  $v_2$  (direction=downwards).

The angular velocity of the wheels can now be calculated using the schematic representation of the relevant velocities, radii and angles shown in Fig. 4.4. When the ball moves in a direction at an angle  $\phi$  relative to forward motion, the wheels describe circles on the surface of the ball. The radii of the circles that the wheels describe on the surface of the ball depend on the angles  $\theta$  and  $\phi$  and the radius of the ball  $r_b$ . From the figure it is easy to see that the projection of these radii on a horizontal surface through the center of the ball is given by:

$$r_{p1} = r_{wb} \sin(\theta + \phi) \quad (4.12)$$

$$r_{p2} = r_{wb} \sin(\theta - \phi), \quad (4.13)$$

Fig. 4.5 shows that  $r_{p1}$  is perpendicular to the height  $h$  and thus

$$r_1 = \sqrt{r_{p1}^2 + h^2} \quad (4.14)$$

and similarly, for the other wheel

$$r_2 = \sqrt{r_{p2}^2 + h^2}. \quad (4.15)$$

These are the actual radii of the circles that the wheels describe on the ball. When no slip occurs, the velocities  $v_1$  and  $v_2$  at the contact points are related to the radii  $r_1$ ,  $r_2$ , the radius of the wheels  $r_w$  and the angular velocities  $\omega_1$ ,  $\omega_2$  and  $\omega_b$  of the two wheels and the ball respectively. These relations are

$$v_1 = \omega_b r_1 = \omega_1 r_w \quad (4.16)$$

$$v_2 = \omega_b r_2 = \omega_2 r_w. \quad (4.17)$$

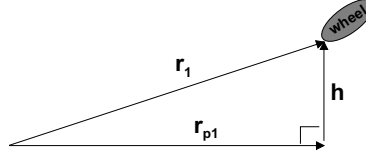


Figure 4.5: Relation between the projected radius  $r_{p1}$ , height  $h$  and the actual radius  $r_1$ .

Combining equations (4.7)-(4.17) and using  $v_b = \omega_b r_b$  gives

$$\omega_1 = \frac{v_b \sqrt{\left(\frac{8}{9} \cos^2 \psi + \frac{1}{3}\right) \sin^2(\theta + \phi) + \frac{8}{9} \sin^2 \psi}}{r_w}, \quad (4.18)$$

$$\omega_2 = \frac{v_b \sqrt{\left(\frac{8}{9} \cos^2 \psi + \frac{1}{3}\right) \sin^2(\theta - \phi) + \frac{8}{9} \sin^2 \psi}}{r_w}, \quad (4.19)$$

Where the angle  $\theta$ , calculated from (4.11), equals 22.2 degrees. The angle  $\psi$  is approximately 30 degrees for the prototype. Substituting this value and taking  $\phi = 0$ , which corresponds to forward motion gives

$$\omega_1 = \omega_2 = 23.1 v_b. \quad (4.20)$$

At certain angles, the wheels have to turn in opposite direction in order to keep contact with the ball. When  $\theta < |\phi| < \pi - \theta$  the wheels will have to turn in opposite direction, and at other angles, the wheels have to turn in the same direction.

The maximum velocity of the wheels occurs when  $\phi = \pm(\frac{\pi}{2} - \theta)$ . Then one of the wheels describes a circle with radius  $r_b$  on the surface of the ball. In this case (4.18) and (4.19) will give  $\omega_1 = \frac{v_b}{r_w}$  or  $\omega_2 = \frac{v_b}{r_w}$ . The maximum velocity will be equal to 1150 RMP for  $v_b = 3m/s$  and  $r_w = 25mm$ . Because the maximum angular velocity of the chosen motors is 5500 RPM, reduction gear modules of 4.3:1 are used to acquire the desired velocity range and to increase the torque of the motors.

The computations assume no slip occurs between the wheels and the ball which is only true when the direction of the trajectory on the ball is exactly orientated along the orientation of the wheels. When the robot drives under a random angle, the wheels will always slip on the ball. Because only forwards and backwards driving is tested in this project, we were able to adapt the orientation of the wheels to match the direction of the trajectory the wheels describe on the ball.

The torque necessary to accelerate the ball can be calculated from

$$T_b = \frac{T_{m1} r_1 + T_{m2} r_2}{r_w}, \quad (4.21)$$

where  $T_{m1}$  and  $T_{m2}$  indicate the torque supplied by motors 1 and 2 and  $T_b$  is the torque on the ball. Notice that the torque delivered to the ball is the sum of the torques delivered by the motors, each multiplied with a gear ratio depending on the radii  $r_1$  and  $r_2$ . When  $|\phi| = \theta$ , all torque has to be delivered by one motor, because the other one will not rotate. The maximum torque that can be delivered constantly by this motor is 27 mNm. With the gear ratio and from (4.21), the maximum torque on the ball is calculated to be 340 mNm. The actual torque needed to accelerate the ball is calculated with (4.2) and is equal to 322 mNm. Even in this extreme case, full acceleration with only one motor, the demanded torque can be supplied.

### 4.2.3 Layout of the TURTLE robot

The layout of the TURTLE robot is shown in Fig. 4.6. The wheels are driven by the motors and the velocity of the motors is measured by the tachometers. The motors are powered by Pulse

Width Modulation (PWM) amplifiers. These amplifiers control the average voltage supplied to the motors by changing the pulse width of the pulses drawn from the 24 Volt batteries. Tachometers are chosen because they are velocity sensors. A tachometer is in fact a inverted motor that generates a voltage proportional to the angular velocity that is applied to this sensor. The tachometers that are used generate 0.52 Volt per 1000 RPM. Additional information can be found on the datasheet, see Appendix A. To be able to detect whether the robot has the ball, a microswitch is attached to each lever. This switch is activated when the lever is pushed upwards by the ball. In the controller design this information will be used for the generation of the reference signal. The motors, tachometers and switches are connected to TUEdACS (TU/e Data Acquisition and Control Systems) units that contains DAC's, ADC's and I/O ports. The TUEdACS units form the connection between the electronics and the laptop that contains the software of the TURTLE robot.

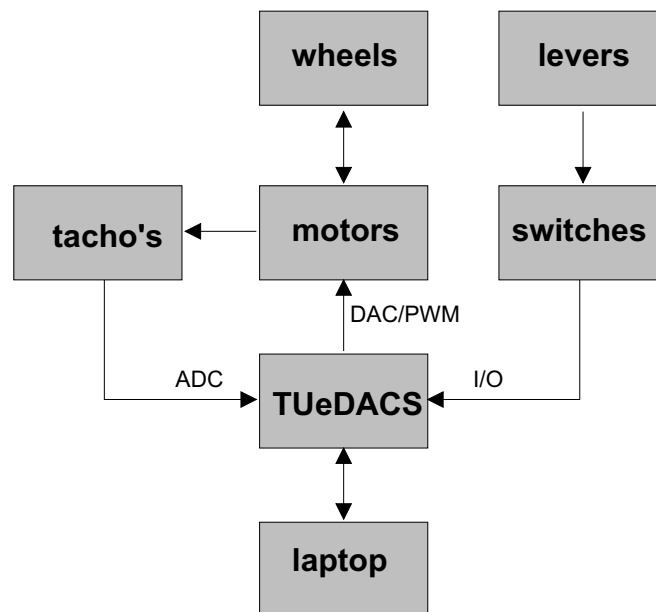


Figure 4.6: The layout of the TURTLE. The wheels are connected to motors that use tachometers to measure velocity. The position of the levers is detected with microswitches. These units are connected to TUEdACS units that form the connection between the electronics and the laptop that contains the software.



## Chapter 5

# Identification and Control

In this chapter, the feedback control design is discussed. A feedback scheme as shown in Fig. 5.1 is used. The controller controls the velocity of the motors but is disturbed by the ball that exerts a force on the wheels.

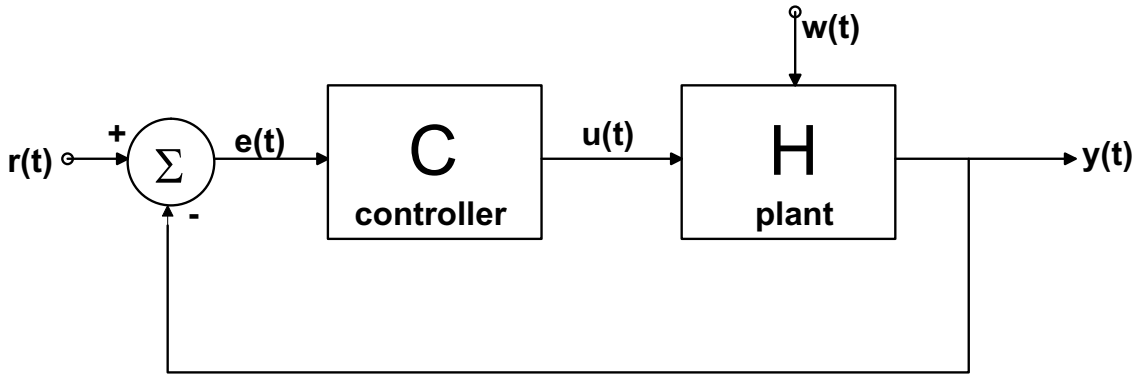


Figure 5.1: Feedback control loop for controller  $C$  and plant  $H$ . The controller gives an input  $u(t)$  to the plant, based on the error  $e(t)$  between the output  $y(t)$  and reference  $r(t)$ . Disturbances e.g. caused by the ball are denoted  $w(t)$ .

First, a system identification of the plant is performed and described in Section 5.1. Section 5.2 discusses the design of the controller. Next, the realization of the desired behavior by means of a reference velocity is discussed in 5.3. Finally in Section 5.4 the implementation of the controller is presented.

### 5.1 System Identification

The frequency response of the system is measured in open loop. A constant voltage plus noise is used as input and the signal of the tachometers is monitored. Fig. 5.2 show the estimated frequency response functions (FRF) that are calculated with the Matlab command “tfe”. Because the system consists of two levers, each with a motor with tachometer, a transmission with toothed pulleys and belt and a wheel, two lines are shown in the figure. Each line represents one side of the system. The main part of the FRF shows a -20 dB per decade slope, due to the inertia of the system. When measuring position, a -40 dB per decade slope is expected for inertia, but because this system measures velocity, a -20 dB per decade slope is expected in this experiment.

Low frequent, the response of the system goes to a constant value. Also, the phase is zero for low frequencies in stead of the -90 that is expected for a pure inertia. This is caused by damping

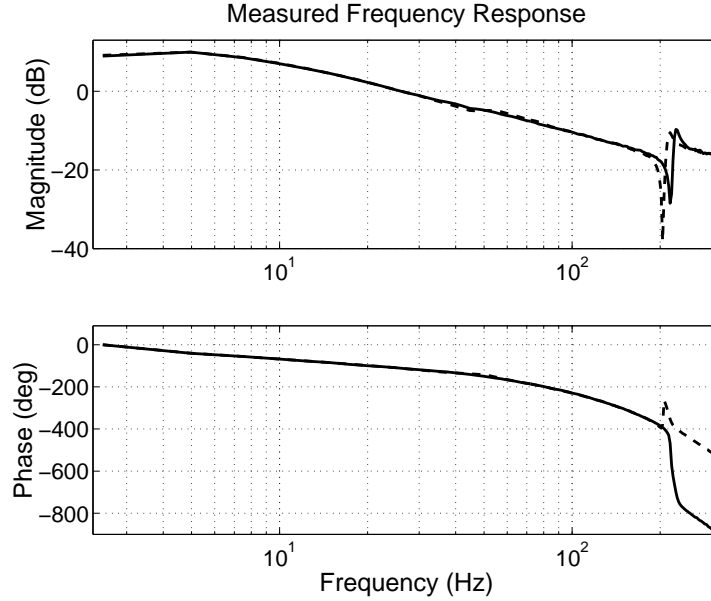


Figure 5.2: Measured frequency response functions of the prototype. Each line represents the response of one of the sides, consisting of a motor with tachometer, a transmission with toothed pulleys and belt and a wheel.

in the system, probably friction at the axis of the wheels. The coherence between the input and output is calculated using the Matlab command “cohere”, see Fig. 5.3. When the coherence is above 0.8, the calculated FRF is reliable. High frequent the coherence is poor, but up til 200 Hz the FRF is reliable. The high frequency resonances are probably caused by the coupling between motor and tachometer. A small, damped resonance, observed at 50 Hz, is probably caused by the coupling between motor and wheel.

The phase plot shows a decrease of the phase with increasing frequencies. This is caused by the delay in the system, due to the digital implementation. This delay can cause instability if the bandwidth is increased to much by the controller.

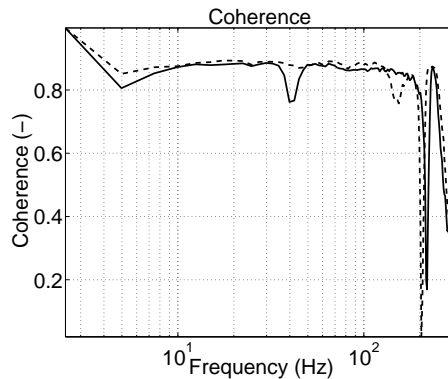


Figure 5.3: Measured coherence functions of the prototype. The coherence is good for low frequencies and poor for frequencies higher than 200 Hz.

## 5.2 Controller Design

Because both sides have almost the same FRF, one controller is designed for both sides. A proportional integral (PI) controller is used to eliminate steady state errors and keep the system stable. The transfer function of the controller is given by

$$C(s) = k_p \frac{s + 2\pi k_i}{s}, \quad (5.1)$$

where  $k_p$  is the proportional gain and  $k_i$  is the pole of the integrating action in Hz. In order to reduce phase lag in the neighborhood of the bandwidth,  $k_i$  is chosen at a low frequency, namely 1 Hz. The proportional gain is increased in order to increase the bandwidth. A phase margin of 60 degrees is chosen to assure stability. This results a  $k_p$  equal to 1. The controlled open loop (CH) and closed loop (CH/1+CH) frequency responses are depicted in Fig. 5.4. Here, H is the measured FRF. The bandwidth of the controlled system is 27 Hz, which will probably be fast enough because the reference will not change that fast. A Nyquist plot, shown in Fig. 5.2, also shows that the system is stable and has a vector margin of approximately 0.5. Fig. 5.2 shows that this results in a sensitivity (1/1+CH) below 6 dB.

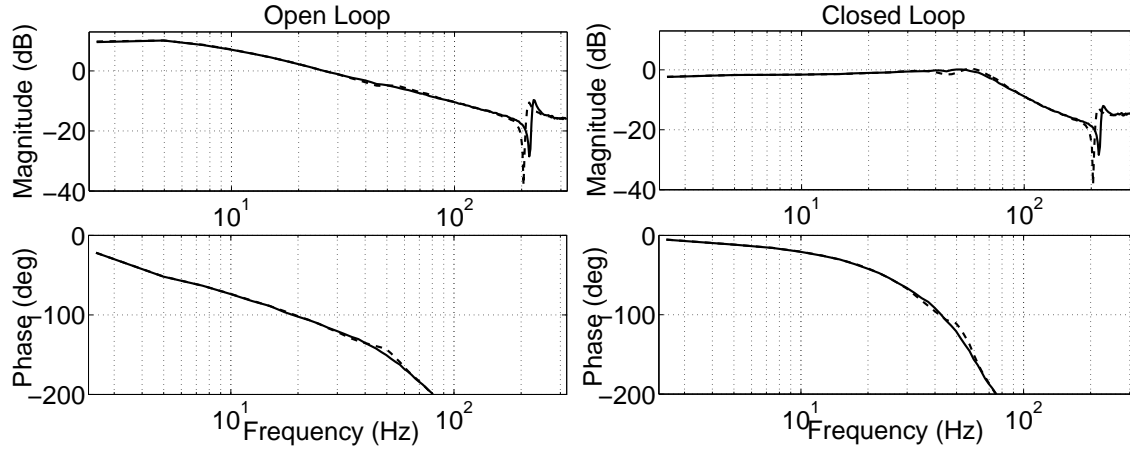


Figure 5.4: Bode diagrams of the system plus controller. Left: open loop (CH), Right: Closed loop (CH/1+CH).

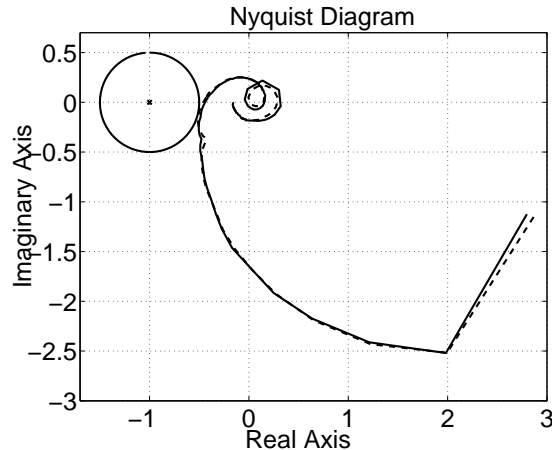


Figure 5.5: Nyquist plot of controlled system. The system has a vector margin of approximately 0.5.

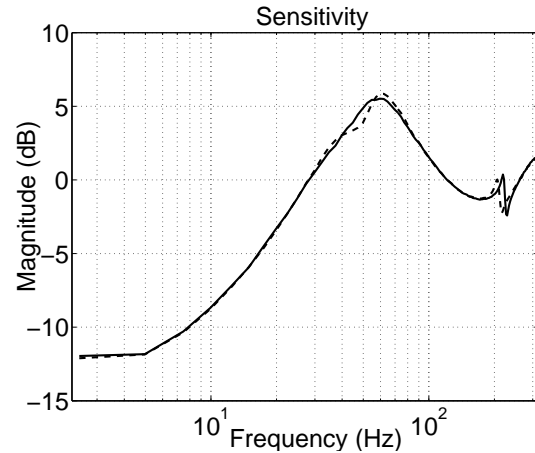


Figure 5.6: Plot of the sensitivity of the system to disturbances (1/1+CH). The peak sensitivity is <6dB.

### 5.3 Reference Velocity Generation

The TURTLE robot has a vision system that is able to detect the position of the ball. Microswitches are used to detect whether or not the ball is between the two levers. The reference velocity is based on the information from these sensors. The controller has to deal with three different situations, namely catching, dribbling and idle mode. Fig. 5.7 shows schematically how the reference velocity is determined. When the robot has the ball, the velocity of the wheels is matched to the velocity of the robot according to the relations derived in Section 4.2.2. When the robot is not in possession of the ball and the vision system of the Turtle robot detects the ball in front of the robot, the wheels rotate towards the robot with constant velocity  $v_{catch}$ . If the ball touches the wheels it is pulled towards the robot, the levers move outwards and activate the microswitches. Then the robot is in dribbling mode. If the robot does not have the ball and the ball is not in front of the robot the wheels are stopped to save energy.

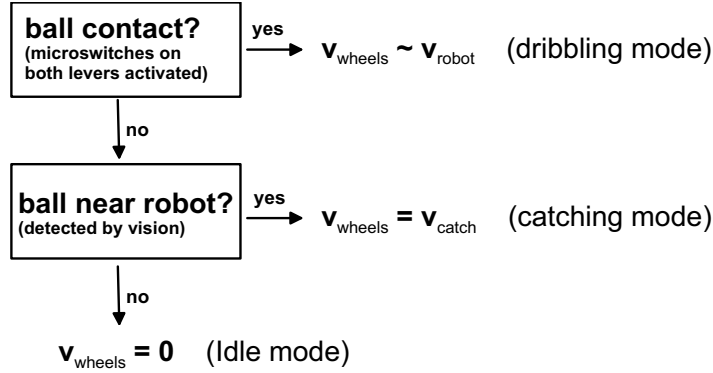


Figure 5.7: Flow scheme of the choice of reference speed of the wheels. The software detects whether the levers are in contact with the ball by means of the microswitches and if not, it checks if the ball is near the robot. With this information a choice for the reference speed is made.

### 5.4 Implementation

Because of the digital environment a digital controller is needed. Tustin discretization combined with an inverse z-transformation is used to transform the continuous time transfer function of the controller. The Tustin transformation yields

$$s = \frac{2}{T} \frac{z - 1}{z + 1} \quad (5.2)$$

where  $s$  is the Laplace variable,  $z$  the complex variable and  $T$  the sample time. This transformation leads to an equation in powers of  $z$  that can be transformed to the actual samples  $\sigma[n]$  using a special case of the inverse z-transformation, namely

$$z^{-k} = \sigma[n - k] \quad (5.3)$$

The TURTLE robot software is based mainly on Matlab Simulink. A S-function is used to implement the PI controller in the software of the robot. A representation of the Simulink block is shown in Fig. 5.8. The reference velocity can be created easily from the inputs with a few conditional statements (if,else,etc.) because of the C programming language. It is created according to the flow scheme discussed in section 5.3. The microswitches are used to detect whether the ball is between the two wheels. The distance and angle to the ball estimated by the vision system are used to detect whether the robot is near the ball. The voltage measured by the tachometers is used to calculate the error. A special tuning input allows real-time adjustment of the catching

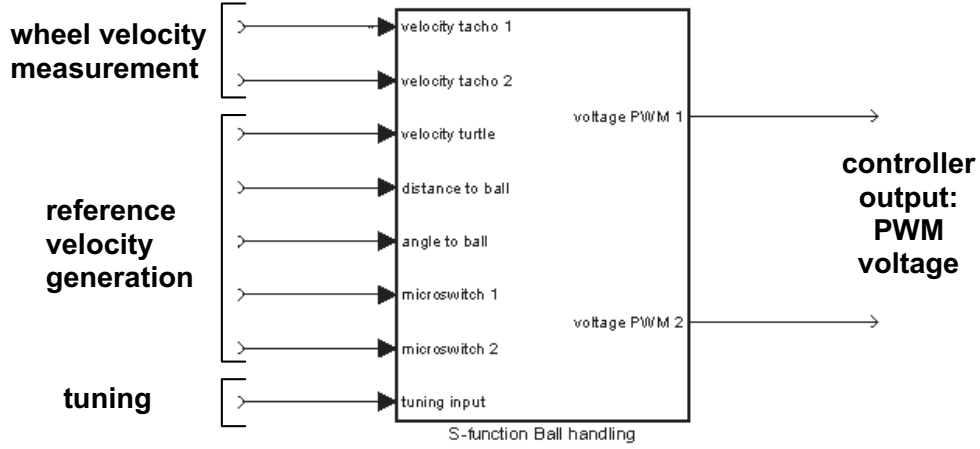


Figure 5.8: Simulink block of the controller. The inputs are used to create the reference velocity and to measure the velocity of the wheels via the tachometers. A special input is the tuning input that allows the tuning of the tachometers.

velocity, and tuning of the reference velocity. This will probably be necessary because of deviations from the theoretical values calculated in Section 4.2.2. Also it could be desirable to decrease the velocity of the wheels a little to push the ball against the robot. To facilitate the controller design, the parameters of the PI controller are programmed as parameters of the s-function. For further details about the implementation see the source code in appendix B.



## Chapter 6

# Performance Evaluation

Only forward and backward motion are tested in this project because of lack of time. However the system could easily be expanded for turning and driving sideways by giving each wheels its own reference velocity according to the relations given in Section 4.2.2.

Experiments have been conducted by driving with the Turtle bot in manual mode using a joystick. The inputs for the distance to the ball and angle to the ball (see Section 5.4) are set to zero to enable testing without the use of the vision of the robot. The tuning was done first. We observed that the velocity of the wheels should be less for forward than for backwards motion. When driving forwards, the velocity of the wheels should be less than that of the ball in order to keep the ball pushed against the robot. When driving in reverse direction the opposite is necessary. Fig. 6.1 illustrates this.

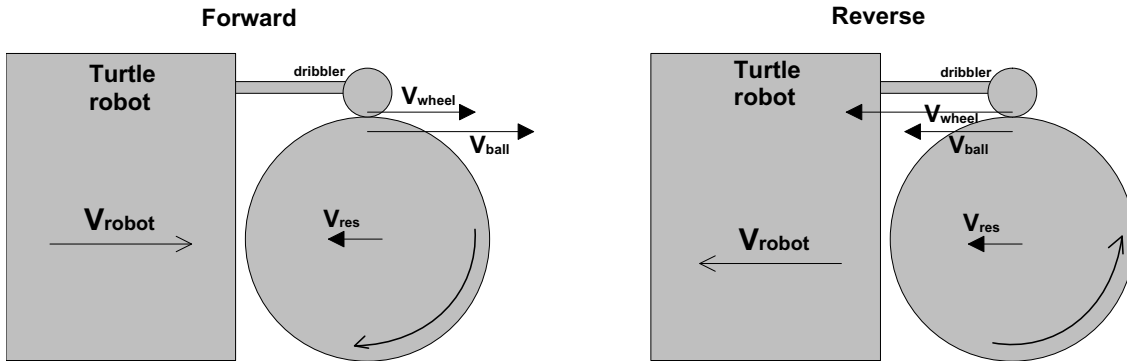


Figure 6.1: Tuning of the reference velocity of the wheels. The velocity of the wheels should be less for forward than for reverse motion. This results in a nett velocity difference, pushing the ball towards the robot.

The tuned gains are given in Table 6.1. Because the voltage of the tacho is used directly in the feedback loop, the tuned values have to be converted to the form  $\omega = gain * v_b$  to be able to compare them to value calculated in (4.20). This can be done using the fact that the tachometer generates 0.52 Volt per 1000 RPM. Note that the tachometer is coupled directly to the axis of the motor, there is no gear reduction between the motor and the tacho. The actual tuned values have a significantly different value than the value calculated in (4.20), which is 23.1. This is partly explained by the statement made above that the gain should be adapted to push the ball against the robot. Also, it was observed that the tires slipped on the rims significantly. Another explanation is that in practice the angles and positions of the wheels always differ from their designed values.

Dribbling with the ball works properly. When driving forwards, backwards, braking and accel-

Parameter	Value	Relation $\omega \leftrightarrow v_b$
forward gain	$V_{ref} = 0.5v_b$	$\omega = 6.4v_b$
backward gain	$V_{ref} = 5.0v_b$	$\omega = 63.5v_b$
catching	$V_{ref} = -0.5V_{olt}$	$\omega = 100rad/s$

Table 6.1: Tuned values of the gains of the controller. A voltage  $V_{ref}$  was used as a reference for the controller. This voltage can be converted in order to compare it to the theoretical relation between angular velocity  $\omega$  and velocity of the ball  $v_b$ .

erating, the ball stays in contact with the robot. This is a great improvement compared to the current situation where the ball moves away from the robot when braking or driving in reverse. The possibility of driving backwards with the ball can be very advantageous in a match situation when an opponent is blocking the way or to pull the ball back when it is near the boundaries of the field. A minor problem occurred with the orientation of the robot. The velocity of the robot is estimated from the encoders of the propulsion wheels. This velocity is expressed in global coordinates and has to be transformed to the local coordinates of the TURTLE robot. Because only forward and backward motion are tested this was no obstruction for this project. For future projects this has to be resolved.

Satisfactory catching ability is also observed. The wheels pull the ball towards the robot, pushing the levers outwards. When the ball is clamped between the levers, the microswitches are activated and the reference velocity is matched to the velocity of the robot as desired. However, when the velocity difference between ball and robot is too large, the ball bounces away. Furthermore, when the ball approaches the robot from an angle, the motor bracket obstructs the contact to the wheel. Even though the prototype was not designed for turning and sideways motion, driving sideways and turning are also possible up to a certain velocity. This is caused by the catching capability and reference velocity generation. When the robot turns or moves sideways, it pushes the ball away from the robot, until the microswitches are turned off due to the return of the levers. Then the robot goes into catching mode pulling the ball back towards the robot until the switches are activated again. Then the ball rolls away again starting this process again.

The Lego wheels proved to be the weakest link in the prototype design. Although the grip of the wheels is perfect, the tires slipped on their rims and ran off the rims constantly. An other undesirable effect was observed in the controller. When a wheel is blocked for a period of time, the integrating action in the controller builds up a large error causing “windup”. When the wheel is free again it unwinds, rotating very fast until the controller stabilizes again.

A photograph of an improved version of the prototype is shown in Fig. 6.2. This prototype has better wheels and a more robust lever design.



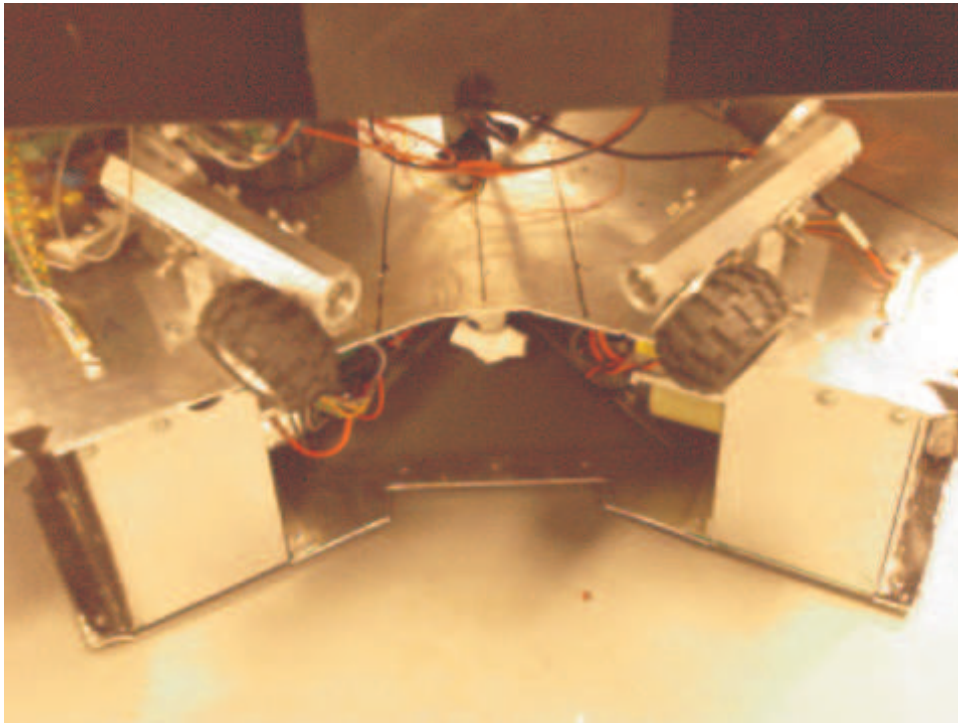


Figure 6.2: Photograph of a improved version of the prototype. This prototype has better wheels and a stronger lever design.



## Chapter 7

# Conclusion and Recommendations

### 7.1 Conclusion

Various concepts for stopping and receiving a soccer ball were compared. The chosen concept uses two wheels placed on levers that touch the side of the ball. The levers exert a force towards the ball to give the wheels good grip on the ball. Catching of the ball can be achieved by pulling it between the two levers by rotating the wheels towards the robot.

A prototype of the design is realized out of metal profiles. The power needed to drive the wheels is supplied by 10 Watt DC motors. The velocities of the wheels relative to the velocity of the robot have been calculated and a PI controller is designed to control the velocity of the wheels. The reference velocity of this wheels is based on the information gathered from the vision system of the TURTLE robot about the position of the ball and two microswitches that detect ball possession. The controller has been implemented in a TURTLE robot and tested for forward and backward motion. The reference velocity is tuned to achieve optimal performance. The tuned values differed from the theoretical values due to differences in the geometry and due to wheel slip. Also the tuned values needed to be adapted to push the ball against the robot.

The dribbling mechanism has shown great potential for the Tech United Robocup team. It enables forward and backward dribbling and keeps the ball against the robot when braking. An approaching ball can be caught by the system. The wheels pull the ball towards the robot when the system is catching the ball. The software can be expanded to enable turning and sideways motion too. Further development of this concept will be of great value to the team in future matches.

### 7.2 Recommendations

The following issues of the catching and dribbling mechanism can be improved:

- **Wheels:** Better wheels that have the tire attached to the rim should be used. Also, the wheels should be mounted in such a way that they are able to catch a ball that approaches from an angle. The motor bracket obstructs this in this design. Because the wheels have to slip when driving in an arbitrary direction, omnidirectional wheels could be used to reduce the wear of the wheels.
- **Levers:** The levers in this design are not strong enough to deal with real match conditions. A final design does not need the adjustment capabilities of this prototype. This can increase robustness greatly. Also the motor and transmission should be well protected which is not the case in this prototype.
- **Mechanical stop:** The ball is pushed towards the robot against an edge of a horizontal plate when dribbling, causing a non smooth dribble. Using a well designed mechanical

stop, possibly a V-shaped opening constructed out of vertical plates can reduce the friction between ball and robot and make dribbling smoother. Then, the force of the ball is exerted on the mechanical stop instead of on the wheels enabling better control over the ball.

- **Force on the ball:** The levers currently exert a force on the ball due to their own weight. The use of suitable springs can be used to optimize the force exerted on the ball. Too much force will push the ball away and too less will not create enough traction with the ball. It also influences the catching capabilities. Lighter levers can probably catch faster balls because they move outwards faster due to their lower inertia. The optimal situation would probably be a light construction with a spring to exert just the right amount of force.
- **Available space:** The top layer of the robot was raised because the mechanism does not fit inside the robot. For the prototype this was not a problem, but a future design should fit inside the robot.
- **Motors:** The motors are quite big, around 120 mm long and 3 mm diameter. The use of smaller motors is probably possible because the current motors have more power than necessary. Measurements of the current drawn from the batteries can provide information about the actual power that is needed.
- **Ball contact detection:** The detection of ball possession is currently discrete, detecting only whether the levers are fully outwards. In reality the wheels make contact to the ball before ball contact is detected. It would be better to detect the position of the ball and adjust the controller signal accordingly. When the ball makes first contact with the wheels must turn fast to grab the ball, but as soon as the ball starts to move towards the robot, the velocity should gradually be reduced until the ball has the same velocity as the robot. The measurement of the ball position can be done with the sonar system that is currently being developed. An other possibility is to use a sensor that can measure the position of the lever continuously. This can be a simple angle sensor consisting of a potentiometer.
- **Software:** The controller can be expanded to be able to deal with turning and driving sideways. Each wheel should have its own reference velocity adjusted to keep the ball with the robot under all circumstances. Anti-windup measures should be taken to ensure stability of the controller. Also the position of the ball could be controlled rather than the velocity of the wheels, using the velocity of the robot as feedforward. This could result in a better dribbling behavior because the controller will keep the ball against the robot.

# Bibliography

- [1] <http://www.Robocup.org>
- [2] David Chung, Akbar Dhanaliwala, Sewan Kim, Jie Dong Leou, Evan Malone, Jeremy Miller, Eryk Brian Nice, Sean R. Richardson, Ken Sterk; *RoboCup Mechanical Engineering Documentation Fall 2001 - Spring 2002*; p.37-54; [http://robocup.mae.cornell.edu/documentation/robocup/2002/\\_Toc9571341](http://robocup.mae.cornell.edu/documentation/robocup/2002/_Toc9571341)
- [3] Steve Stancliff; *Evolution of Active Dribbling Mechanisms in RoboCup*; April 25, 2005; [http://www.cs.cmu.edu/afs/cs/academic/class/16741-s06/www/Projects/stancliff\\_16741.pdf](http://www.cs.cmu.edu/afs/cs/academic/class/16741-s06/www/Projects/stancliff_16741.pdf)
- [4] Concepts of Bar Dribblers available at <http://robocup.mae.cornell.edu/>
- [5] D. Ball, G. Wyeth, D. Cusack; *Design of a Team of Soccer Playing Robots*; [http://www.itee.uq.edu.au/dball/roboros/publications/roboros\\_AMIRE03.pdf](http://www.itee.uq.edu.au/dball/roboros/publications/roboros_AMIRE03.pdf)
- [6] "The Cornell RoboCup Team," R. D'Andrea, T. Kalmar-Nagy, P. Ganguly, M. Babish; *RoboCup 2000: Robot Soccer World Cup IV*, P. Stone, T. Balch and G. Kraetzschmar, Eds., New York: Springer, 2002, pp. 41-51.




## Appendix A

# Datasheets DC Motor and Tachometer

**RE 25** Ø25 mm, Precious Metal Brushes CLL, 10 Watt, CE approved



**M 1:2**

-  Stock program  
 Standard program  
 Special program (on request)

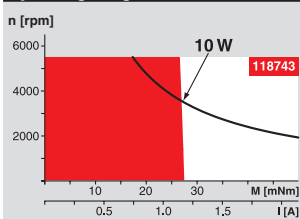
**Order Number**

		118740	118741	118742	118743	118744	118745	118746	118747	118748									
Motor Data																			
Values at nominal voltage																			
1	Nominal voltage	V	4.5	8.0	9.0	12.0	15.0	18.0	24.0	32.0	48.0								
2	No load speed	rpm	5350	5310	5230	4850	4980	4780	5190	5510	5070								
3	No load current	mA	79.6	44.3	38.6	26.2	21.7	17.2	14.3	11.6	6.95								
4	Nominal speed	rpm	4910	4510	4230	3820	3940	3740	4150	4470	4030								
5	Nominal torque (max. continuous torque)	mNm	11.4	20.2	25.1	28.8	28.8	28.8	28.8	28.8	28.7								
6	Nominal current (max. continuous current)	A	1.50	1.50	1.50	1.26	1.03	0.823	0.668	0.529	0.325								
7	Stall torque	mNm	138	139	126	137	138	133	144	152	140								
8	Starting current	A	17.2	9.73	7.72	5.82	4.83	3.72	3.28	2.76	1.56								
9	Max. efficiency	%	87	87	86	87	87	87	87	88	87								
Characteristics																			
10	Terminal resistance	Ω	0.261	0.822	1.17	2.06	3.10	4.84	7.31	11.6	30.9								
11	Terminal inductance	mH	0.0275	0.0882	0.115	0.238	0.353	0.551	0.832	1.31	3.48								
12	Torque constant	mNm / A	8.00	14.3	16.4	23.5	28.6	35.8	44.0	55.2	90.0								
13	Speed constant	rpm / V	1190	667	584	406	333	267	217	173	106								
14	Speed / torque gradient	rpm / mNm	39.0	38.3	41.6	35.6	36.1	36.0	36.1	36.3	36.4								
15	Mechanical time constant	ms	4.74	4.15	4.12	4.00	3.98	3.97	3.97	3.97	3.97								
16	Rotor inertia	qcm <sup>2</sup>	11.6	10.3	9.45	10.7	10.5	10.5	10.5	10.4	10.4								

## Specifications

- |                                       |   |                |
|---------------------------------------|---|----------------|
| <b>Thermal data</b>                   |   |                |
| 17                                    | Thermal resistance housing-ambient                              | 14 K/W         |
| 18                                    | Thermal resistance winding-housing                              | 3.1 K/W        |
| 19                                    | Thermal time constant winding                                   | 124 s          |
| 20                                    | Thermal time constant motor                                     | 910 s          |
| 21                                    | Ambient temperature   | -20 ... +85°C  |
| 22                                    | Max. permissible winding temperature                            | +100°C         |
| <b>Mechanical data (ball bearing)</b> |   |                |
| 23                                    | Max. permissible speed  | 5500 rpm       |
| 24                                    | axial play  | 0.05 - 0.15 mm |
| 25                                    | Radial play   | 0.025 mm       |
| 26                                    | Max. axial load (dynamic)                                       | 3.2 N          |
| 27                                    | Max. force for press fits (static)<br>(static, shaft supported) | 6.3 N<br>270 N |
| 28                                    | Max. radial loading, 5 mm from flange                           | 16 N           |

### Operating Range



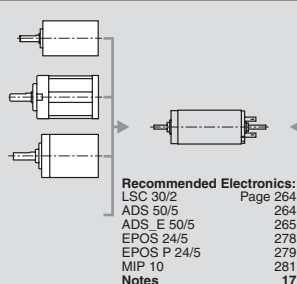
## Comments

- Continuous operation**  
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.  
= Thermal limit.
- Short term operation**  
The motor may be briefly overloaded (recurring).
- Assigned power rating**

**maxon Modular System**

- |                             |                               |       |
|-----------------------------|-------------------------------|-------|
| <b>Other specifications</b> |                               |       |
| 29                          | Number of pole pairs          | 1     |
| 30                          | Number of commutator segments | 11    |
| 31                          | Weight of motor               | 130 g |
| CLL = Capacitor Long Life   |                               |       |
- Values listed in the table are nominal.  
Explanation of the figures on page 49.
- Option**  
Preload ball bearings    Preload strength min. 3.2 N

- Planetary Gearhead**  
 Ø26 mm  
 0.5 - 2.0 Nm  
 Page 224
- Planetary Gearhead**  
 Ø32 mm  
 0.4 - 2.0 Nm  
 Page 226
- Planetary Gearhead**  
 Ø32 mm  
 0.75 - 6.0 Nm  
 Page 227 / 229

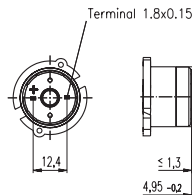


Overview on page 17 - 21

- Encoder MR**  
128 - 1000 CPT,  
3 channels  
Page 246
- Encoder Enc**  
Ø22 mm  
100 CPT, 2 channels  
Page 248
- Encoder HED... 5540**  
500 CPT,  
3 channels  
Page 250 / 252
- DC-Tacho DCT**  
Ø22 mm  
0.52 V  
Page 259



## DC-Tacho DCT 22, 0.52 Volt



### Important Information

- Tacho with moving coil, maxon system.
- Tacho with precious metal commutation.
- To establish total inertia add motor and tacho inertias.
- With the output shaft turning CW as seen from the mounting surface, the tacho output voltage will be positive at the + terminal.
- A high impedance load is recommended at tacho terminals.
- The tacho current should be kept low.
- The indicated resonance frequency refers to the motor-tacho rotor system.

maxon tacho

- ☒ Stock program  
☐ Standard program  
☐ Special program (on request)

### Order Number

118908 118909 118910

### Type

Shaft diameter (mm)	2	3	4



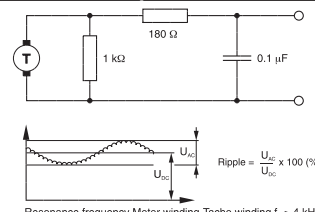
### Combination

+ Motor	Page	+ Gearhead	Page	Overall length [mm] / see: + Gearhead
RE 25, 10 W	77	GP 26, 0.5 - 2.0 Nm	224	76.8
RE 25, 10 W	77	GP 32, 0.4 - 2.0 Nm	226	•
RE 25, 10 W	77	GP 32, 0.75 - 4.5 Nm	227	•
RE 25, 10 W	77	GP 32, 1.0 - 6.0 Nm	229	•
RE 25, 20 W	78			76.8
RE 25, 20 W	78	GP 26, 0.5 - 2.0 Nm	224	•
RE 25, 20 W	78	GP 32, 0.4 - 2.0 Nm	226	•
RE 25, 20 W	78	GP 32, 0.75 - 4.5 Nm	227	•
RE 25, 20 W	78	GP 32, 1.0 - 6.0 Nm	229	•
RE 26, 18 W	79			79.8
RE 26, 18 W	79	GP 26, 0.5 - 2.0 Nm	224	•
RE 26, 18 W	79	GP 32, 0.4 - 2.0 Nm	226	•
RE 26, 18 W	79	GP 32, 0.75 - 4.5 Nm	227	•
RE 26, 18 W	79	GP 32, 1.0 - 6.0 Nm	229	•
RE 35, 90 W	81			89.0
RE 35, 90 W	81	GP 32, 0.75 - 4.5 Nm	228	•
RE 35, 90 W	81	GP 32, 1.0 - 6.0 Nm	229	•
RE 35, 90 W	81	GP 42, 3.0 - 15 Nm	232	•
RE 36, 70 W	82			89.3
RE 36, 70 W	82	GP 32, 0.4 - 2.0 Nm	226	•
RE 36, 70 W	82	GP 32, 0.75 - 4.5 Nm	228	•
RE 36, 70 W	82	GP 32, 1.0 - 6.0 Nm	229	•
RE 36, 70 W	82	GP 42, 3.0 - 15 Nm	232	•

### Technical Data

Output voltage per 1000 rpm	0.52 V	Max. recommended current	10 mA
Terminal resistance tacho	56.6 Ω	Tolerance of the output voltage	± 15 %
Typical peak to peak ripple	≤ 6 %	Rotor inertia (tacho only)	< 3 gcm <sup>2</sup>
Ripple frequency per turn	14	Resonance frequency with motors on p. 77 - 79	> 2 kHz
Linearity between 500 and 5000 rpm unloaded	± 0.2 %	with motors on pages 86, 88	> 3 kHz
Linearity with 10 kΩ load resistance	± 0.7 %	with motors on pages 81, 82	> 4.5 kHz
Reversal error	± 0.1 %	Temperature range	-20 ... +65°C
Temperature coefficient of EMF (magnet)	-0.02 % / °C	Option: Pigtailed in place of solder terminals.	
Temperature coefficient of coil resistance	+0.4 % / °C		

### Connection example





## Appendix B

# Source Code Controller

```
/*
 * File : dribblewheel2.c
 *
 * Abstract:
 *     Drives the dribble wheels for the ball handling of the TURTLE bots.
 *
 * Inputs:
 *     0: Velocity tacho 0
 *     1: Velocity tacho 1
 *     2: Velocity TURTLE bot (smooth reference)
 *     3: Distance to ball
 *     4: Angle to ball
 *     5: Ball contact sensor 0
 *     6: Ball contact sensor 1
 *     7: Tunable parameter input: gain tacho's
 *
 * Outputs:
 *     0: Control signal DAC 0
 *     1: Control signal DAC 1
 *     2: Smooth reference signal for wheel velocity
 *
 * Parameters: freq_LP kp0 ki0 kp1 ki1 vel_red vel_catch
 *
 *
 * Copyright 2006 R.J.E. Merry, R. Hoogendijk.
 * Version: 1.0
 */

#define S_FUNCTION_NAME  dribblewheel2
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include "dribblewheel2.h"

struct BallParams_t{
    real_T GainTach;
};

/*=====*
```

```
* Build checking *
*=====*/

/* Function: mdlInitializeSizes =====*/
static void mdlInitializeSizes(SimStruct *S)
{
    int_T rwrk_size, i;

    /* S-function parameters field of the dialog box empty */
    ssSetNumSFcnParams(S, 7);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    /* Input Ports */
    if (!ssSetNumInputPorts(S, NINPUTS)) return;
    for (i=0;i<NINPUTS;i++) {
        ssSetInputPortWidth(S, i, 1);
        /* Input port width */
        ssSetInputPortDirectFeedThrough(S, i, NEEDSINP);
        /* Direct feedthrough if the input is used in either the mdlOutputs or
        mdlGetTimeOfNextVarHit functions. */
        ssSetInputPortRequiredContiguous(S, i, true);
        /* Specifies that the signal elements entering the specified port must
        occupy contiguous areas of memory. This allows a method to access the
        elements of the signal simply by incrementing the signal pointer
        returned by ssGetInputPortSignal. */
    }

    /* Output ports */
    if (!ssSetNumOutputPorts(S, NOUTPUTS)) return;
    for (i=0;i<NOUTPUTS;i++) {
        ssSetOutputPortWidth(S, i, 1);
    }

    /* workspace */
    rwrk_size=sizeof(pi_par)/sizeof(real_T)+1;
    ssSetNumRWork(S,rwrk_size);

    /* The number of sample times the s-function has */
    ssSetNumSampleTimes(S, 1);
}

/* Function: mdlInitializeSampleTimes =====*/
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}
```

```

/* Function: mdlInitializeConditions =====*/
#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
static void mdlInitializeConditions(SimStruct *S)
{
    real_T *prwrk=ssGetRWork(S);
    ppi_par ppar;

    ppar=(ppi_par) prwrk;

    ppar->kp0 = *mxGetPr(ssGetSFcnParam(S,1));
    /* Gain controller C = kp*((s+2*pi*ki)/s) LEFT SIDE*/
    ppar->ki0 = *mxGetPr(ssGetSFcnParam(S,2));
    /* Frequency where the I action stops [Hz] LEFT SIDE*/
    ppar->kp1 = *mxGetPr(ssGetSFcnParam(S,3));
    /* Gain controller C = kp*((s+2*pi*ki)/s) RIGHT SIDE*/
    ppar->ki1 = *mxGetPr(ssGetSFcnParam(S,4));
    /* Frequency where the I action stops [Hz] RIGHT SIDE*/
    ppar->freq_LP = *mxGetPr(ssGetSFcnParam(S,0));
    /* Cut-off frequency LP filter to smooth reference [Hz] */
    ppar->vel_red = *mxGetPr(ssGetSFcnParam(S,5));
    /* Reduction factor velocity ball with respect to velocity robot */
    ppar->vel_catch = *mxGetPr(ssGetSFcnParam(S,6));
    /* Velocity for catching the ball relative to robot [m/s]??*/
    ppar->balldist_turn = 1.0;
    /* Distance of ball to robot where the bands start to turn [m] */
    /*tunable ppar->velampl_band = *mxGetPr(ssGetSFcnParam(S,0));
    /* Amplification factor for velocity of robot to account for
    gear/contactpoint ratio/tacho conversion factor*/
    ppar->band_turnifangle = PI/2;
    /* Angle of ball to robot below which the bands start to turn [rad] */
    ppar->preverror0 = 0.0;
    /* Initialize variables to save previous data */
    ppar->preverror1 = 0.0;
    ppar->prevvoltage0 = 0.0;
    ppar->prevvoltage1 = 0.0;
    ppar->prev_vin = 0.0;
    ppar->pprev_vin = 0.0;
    ppar->prev_vout = 0.0;
    ppar->pprev_vout = 0.0;
    ppar->step_size = 0.0;
}
#endif

/* Function: td_controller LEFT =====*/
double td_pi0(double error, double preverror, double prevvoltage, SimStruct *S)
{
    real_T *prwrk=ssGetRWork(S);
    ppi_par ppar;

    ppar=(ppi_par) prwrk;

    return ppar->kp0*(1+PI*ppar->ki0*ppar->step_size)*error

```

```
+ ppar->kp0*(-1+PI*ppar->ki0*ppar->step_size)*preverror + prevvoltage;
}

/* Function: td_controller RIGHT =====*/
double td_pi1(double error, double preverror, double prevvoltage, SimStruct *S)
{
    real_T *prwrk=ssGetRWork(S);
    ppi_par ppar;

    ppar=(ppi_par) prwrk;

    return ppar->kp1*(1+PI*ppar->ki1*ppar->step_size)*error
    + ppar->kp1*(-1+PI*ppar->ki1*ppar->step_size)*preverror + prevvoltage;
}

/* Function: td_lowpass =====*/
double td_lowpass(double ref, double prevref, double pprevref, double prevout,
double pprevout, SimStruct *S)
{
    real_T *prwrk=ssGetRWork(S);
    ppi_par ppar;
    real_T om, cA, cB, cC, cD, cE, cF;

    ppar=(ppi_par) prwrk;

    om = 2*PI*ppar->freq_LP;
    cA = om*om*ppar->step_size*ppar->step_size;
    cB = 2*om*om*ppar->step_size*ppar->step_size;
    cC = om*om*ppar->step_size*ppar->step_size;
    cD = 4+om*om*ppar->step_size*ppar->step_size+4*om*ppar->step_size;
    cE = -8+2*om*om*ppar->step_size*ppar->step_size;
    cF = -4*om*ppar->step_size+4*om*om*ppar->step_size*ppar->step_size;

    return cA/cD*ref + cB/cD*prevref + cC/cD*pprevref - cE/cD*prevout - cF/cD*pprevout;
}

/* Function: mdlOutputs =====*/
static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T *prwrk=ssGetRWork(S);
    real_T error0, error1, voltage0, voltage1;
    static real_T v_wheels, v_wheels_smooth;

    /* Retrieve and define input signals */
    const real_T *vel_tacho0 = (const real_T*) ssGetInputPortSignal(S,0);
    const real_T *vel_tacho1 = (const real_T*) ssGetInputPortSignal(S,1);
    const real_T *vel_des = (const real_T*) ssGetInputPortSignal(S,2);
    const real_T *balldist = (const real_T*) ssGetInputPortSignal(S,3);
    const real_T *ballangle = (const real_T*) ssGetInputPortSignal(S,4);
    const real_T *contact_sens0 = (const real_T*) ssGetInputPortSignal(S,5);
    const real_T *contact_sens1 = (const real_T*) ssGetInputPortSignal(S,6);
    const real_T *BParamsIn = (const real_T*) ssGetInputPortSignal(S,7);
```

```
struct BallParams_t BParams;
BParams.GainTachf = BParamsIn[0]; /*forward gain*/
BParams.GainTachb = BParamsIn[1]; /*backward gain*/
BParams.GainTachc = BParamsIn[2]; /*catching velocity*/

/* Define output signals */
real_T *dac0 = (real_T *) ssGetOutputPortSignal(S,0);
real_T *dac1 = (real_T *) ssGetOutputPortSignal(S,1);
real_T *dac2 = (real_T *) ssGetOutputPortSignal(S,2);

ppi_par ppar;

ppar=(ppi_par) prwrk;

/* Retrieve sample time */
ppar->step_size = ssGetStepSize(S);

/* Make reference signal SAME FOR BOTH WHEELS*/

if(*contact_sens0>0.0 && *contact_sens1>0.0)
{
    if(*vel_des>0)
    {
        v_wheels = *vel_des * BParams.GainTachf;
    }
    else
    {
        v_wheels = *vel_des * BParams.GainTachb;
    }
}

else
{
    if(*balldist<ppar->balldist_turn && *ballangle>-ppar->band_turnifangle&&
    *ballangle<ppar->band_turnifangle)
    {
        v_wheels = -1 * (BParams.GainTachc);
    }
    else
    {
        v_wheels = 0;
    }
}

v_wheels_smooth = td_lowpass(v_wheels, ppar->prev_vin, ppar->pprev_vin,
ppar->prev_vout, ppar->pprev_vout, S);

error0 = v_wheels_smooth - *vel_tacho0;
```

```
error1 = v_wheels_smooth + *vel_tacho1;

voltage0 = td_pi0(error0, ppar->preverror0, ppar->prevvoltage0, S);
voltage1 = td_pi1(error1, ppar->preverror1, ppar->prevvoltage1, S);

ppar->prevvoltage0 = voltage0;
ppar->prevvoltage1 = voltage1;
ppar->preverror0    = error0;
ppar->preverror1    = error1;
ppar->pprev_vin     = ppar->prev_vin;
ppar->prev_vin      = v_wheels;
ppar->pprev_vout     = ppar->prev_vout;
ppar->prev_vout      = v_wheels_smooth;

*dac0 = voltage0;
*dac1 = -voltage1;
*dac2 = v_wheels_smooth;

}

/* Function: mdlTerminate =====*/

static void mdlTerminate(SimStruct *S)
{
}

#ifdef MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif

/*header file dribblewheel2.h*/

#define PI 3.14159265358979

#define NINPUTS      8
#define NOUTPUTS     3
#define NEEDSINP     1

typedef struct tag_pi_par {
    real_T kp0;
    real_T ki0;
    real_T kp1;
    real_T ki1;
    real_T freq_LP;
    real_T vel_red;
    real_T vel_catch;
    real_T balldist_turn;
    /* real_T velampl_band;*/
    real_T band_turnifangle;
    real_T preverror0;
}
```



```
    real_T preverror1;  
    real_T prevvoltage0;  
    real_T prevvoltage1;  
    real_T prev_vin;  
    real_T pprev_vin;  
    real_T prev_vout;  
    real_T pprev_vout;  
    real_T step_size;  
} pi_par, *ppi_par;
```