University of Twente

EEMCS / Electrical Engineering
Control Engineering

Self-Balancing Robot 'Dirk'

Stefan Groothuis

Individual Design Assignment

Supervisors:
prof.dr.ir. J. van Amerongen
dr.ir. P.C. Breedveld
dr.ir. J.F. Broenink

June 2008

Report nr. 017CE2008
Control Engineering
EE-Math-CS
University of Twente
P.O.Box 217
7500 AE Enschede
The Netherlands

Summary

In this project, a self-balancing robot was designed and realized, which can be used as a demonstration model.

The kinematics of the system were analyzed and a bond graph model of the physical system was made. Using this model, a continuous PID-controller on the angle of the bar was designed, which was discretized by the bilinear transform to be able to implement the controller in an embedded system. To measure the angle, a gyroscope and an accelerometer were used. These separate measurements were combined by a filter, which compensates a drift effect that occurs when the gyroscope output is integrated. An embedded system was designed using a microcontroller in which the PID-controller and this filter were implemented. The motor is driven by a PWM-signal from the microcontroller and everything can be powered by either six AA-batteries or an external 9 V supply. A lightweight and rigid aluminium construction was made, to which all subsystems are connected.

The project resulted in a working self-balancing robot, as well as a system model which can be used for future work. However, certain aspects could have been done better. The most important aspect is that the power supply for the microcontroller and sensors when using batteries, is not stable when much current is drawn by the motor. A voltage drop in the outputs of the sensors can be measured, which influences the angle measurement and thus the balancing performance of the system. A separate supply for the microcontroller and the sensors should solve the problem.

Samenvatting

Tijdens dit project is een zelfbalancerende robot ontworpen en gerealiseerd, die gebruikt kan worden als een demonstratiemodel.

De kinematica van het systeem is geanalyseerd en een bondgraaf-model van het fysische systeem is opgesteld. Een continue PID-regelaar op de hoek van de staaf is ontworpen, die gediscretiseerd is met behulp van de bilineaire transformatie. Hierdoor kan de regelaar geïmplementeerd worden in een embedded systeem. Een gyroscoop en een accelerometer zijn gebruikt om de hoek te meten. Deze afzonderlijke metingen zijn gecombineerd door middel van een filter, die het drifteffect compenseert, dat ontstaat wanneer de uitgang van de gyroscoop wordt geïntegreerd. Een embedded systeem met een microcontroller is ontworpen, waarin de PID-regelaar en het filter zijn geïmplementeerd. De motor wordt aangestuurd via een PWM-signaal vanuit de microcontroller en alles wordt gevoed uit zes AA-batterijen, of een externe 9 V voeding. Een lichte en stijve constructie is gemaakt, waaraan alle subsystemen gekoppeld zijn.

Het project heeft uiteindelijk geresulteerd in een werkende, zelfbalancerende robot, alsmede een model van het systeem dat gebruikt kan worden als startpunt voor verder onderzoek. Er zijn aspecten die op een betere manier gedaan hadden kunnen worden. Het belangrijkste is, dat de voedingsspanning voor de microcontroller en de sensoren bij gebruik van batterijen, niet stabiel is als er veel stroom door de motor getrokken wordt. Een spanningsdaling van de uitgangen van de sensoren is te meten, wat de hoekmeting en dus de systeemprestaties beïnvloedt. Een aparte voeding voor de microcontroller en de sensoren zal het probleem oplossen.

Contents

1 Introduction **1**
- 1.1 Project Description . . . 1
- 1.2 Report Outline . . . 2

2 System Model **3**
- 2.1 System Specifications . . . 3
- 2.2 Modeling . . . 3
  - 2.2.1 Kinematics . . . 3
  - 2.2.2 Bond graph model . . . 4
- 2.3 Plant Simulations . . . 7
- 2.4 Conclusion . . . 7

3 Controller Design **9**
- 3.1 Criterion and Specifications . . . 9
- 3.2 Continuous Controller . . . 10
- 3.3 Discretizing . . . 12
- 3.4 Conclusion . . . 13

4 System Design and Realization **15**
- 4.1 Embedded System Design . . . 15
  - 4.1.1 Peripheral usage . . . 16
  - 4.1.2 Implementation . . . 17
- 4.2 Angle Measurement . . . 17
  - 4.2.1 Sensors . . . 17
  - 4.2.2 Drift compensation filter . . . 19
  - 4.2.3 Performance . . . 20
- 4.3 Electronics . . . 21
  - 4.3.1 Power supply . . . 21
  - 4.3.2 Motor and motor driver . . . 22
  - 4.3.3 Additional electronics . . . 22
- 4.4 Mechanical Construction . . . 22
- 4.5 System Simulation . . . 23
- 4.6 Total Costs . . . 24
- 4.7 Conclusion . . . 25

5 Results **27**
- 5.1 Achievements . . . 27
- 5.2 System Performance . . . 27
  - 5.2.1 Stability and disturbance compensation . . . 27
  - 5.2.2 Time of operation . . . 27
- 5.3 Reflection . . . 28

6 Conclusion 31
6.1 Conclusions 31
6.2 Design Reconsiderations 31
6.3 System Improvements 32

A Introduction to Planar Mechanics 33

B Diagrams and Schematics 37

C Quick Reference Manual 41
C.1 Features 41
C.2 Operation 41

References 43

1

Introduction

1.1 Project Description

During the Mechatronics-project [2] in the second year, each participating group designs a mechatronic system of their own choice. From this year, however, each group has to build a self-balancing robot during the project. A self-balancing robot was designed in this project as well, as a test case, to encounter possible difficulties. The robot is called ‘Dirk’, which has no further meaning.

A simplified side view of the system is shown in Figure 1.1. It consists of two wheels connected by an axis and a bar connected to that axis in a way it can rotate around it. The bar should be balanced above the wheels and is in an equilibrium position if the center of gravity of the bar is positioned directly above the axis. If the bar moves out of its equilibrium position (1), the center of gravity of the bar is no longer positioned directly above the axis. To compensate for this situation, the wheels have to be moved in the same direction (2) so the axis will be positioned underneath the center of gravity of the bar. This is done by a motor. Thus, the goal is to keep the bar in the equilibrium position by moving the wheels.

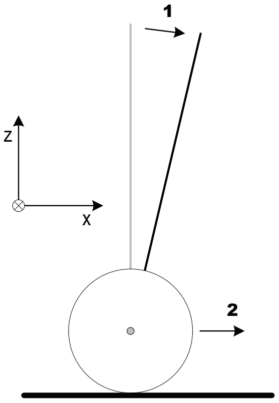

Figure 1.1: Simplified side-view of the system

While designing this mechatronic system, the standard methodology is followed. A model of the physical system is made to understand its dynamic behavior, which is verified using simulations. A controller is then designed, which is tested and verified in simulations using the model. Finally, an embedded system, electronics and mechanics are designed and the system is realized. The result is then evaluated.

The aim of the project was to have a working self-balancing robot, which can be used as a demonstration model.

1.2 Report Outline

In Chapter 2, a bond graph model of the physical system is presented. Chapter 3 presents the controller design. Chapter 4 describes the design and realization of the embedded system, electronics and mechanics. The final results that are achieved are presented in Chapter 5. Conclusions can be found in Chapter 6, as well as design choice reconsiderations and improvements to the system. Appendix A gives an introduction to modeling planar mechanics. A software diagram and electronic and mechanical schematics are presented in Appendix B. Appendix C gives a quick reference manual on how to operate the system.

2

System Model

The modeling of the dynamics of the physical system is needed to understand its behavior. In Section 2.1, system specifications are presented. A bond graph model of the system was made in 20-Sim [3], which is described in Section 2.2. This model was tested in simulations, see Section 2.3.

2.1 System Specifications

The following design and performance specifications were made, while it was kept in mind that the system needs to be used as a demonstration model.

- The system has to be standalone, i.e. everything needed to run the robot has to be integrated onboard;
- The system must be easy to use and must be built robustly;
- The system has to be powered by standard batteries, so they can be replaced easily;
- The system has to function on a flat, non-slippery floor;
- After a little push, the system has to balance itself again;
- The maximum angle of the bar out of the equilibrium position at which the system still has to balance itself, has been chosen 0.175 rad, or 10° (0.35 rad total rotation).

2.2 Modeling

2.2.1 Kinematics

Stabilizing the unstable bar in the x-direction is the essential problem. Due to this point of view, the sideward movements in the y-direction were neglected, resulting in a planar (two-dimensional) mechanical system. Appendix A gives an introduction to modeling planar mechanical systems.

A detailed drawing of the kinematics of the system is shown in Figure 2.1.
The upper rigid body models the bar, the lower rigid body models the wheels. The bar is connected in a joint which allows in-plane rotation around the wheels at a distance $\frac{1}{2}l_{bar}$. The wheels are constraint to a rotation in the xz-plane and are furthermore constraint to a translation in the x-direction. The velocity of the bar in the x-direction, i.e. v_{x-bar}, is the sum of the velocity of the wheels $v_{x-wheels}$ and the x-velocity of the bar due to a rotation $v_{x-bar-rotation}$.

$$v_{x-bar} = v_{x-wheels} + v_{x-bar-rotation} \tag{2.1}$$

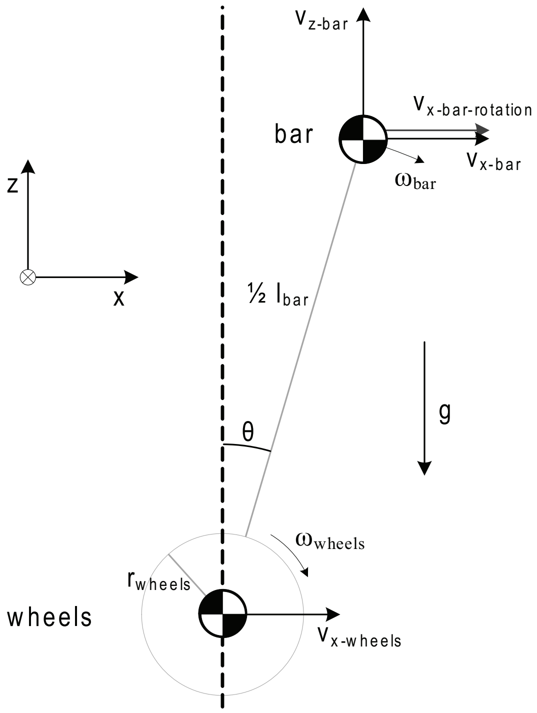

Figure 2.1: Rigid bodies description of the system

The joint represents the motor: the axis of the motor, to which the wheels are connected, rotates in the xz-plane with respect to the housing of the motor, to which the bar is connected. Thus, the angular velocity of the motor ω_{motor} is in fact the difference in angular velocity of the wheels ω_{wheels} and the bar ω_{bar}.

$$\omega_{motor} = \omega_{wheels} - \omega_{bar} \tag{2.2}$$

These two relations are the coupling between the three submodels, i.e. the bar, the wheels and the motor.

2.2.2 Bond graph model

The two rigid bodies and the joint were constructed in 20-Sim using bond graphs. Figure 2.2 shows the bond graph model of the physical system, i.e. the plant.

Bar

Two mechanical domains are present, i.e. the rotation and the translation domain. The corresponding flow variables are the angular velocity ω_{bar} and the translational velocity v_{bar} in x and z-direction, respectively. The translation domain is found by a transformation of the rotation domain using the variable angle θ, which is the integral of the angular velocity ω_{bar}. This transformation is done by two MTF-elements; one for the x-direction and one for the z-direction. The flow variables are related by:

$$\begin{bmatrix} v_{x-bar} \\ v_{z-bar} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} l_{bar} \cdot cos\,\theta \\ -\frac{1}{2} l_{bar} \cdot sin\,\theta \end{bmatrix} \cdot \omega_{bar} \tag{2.3}$$

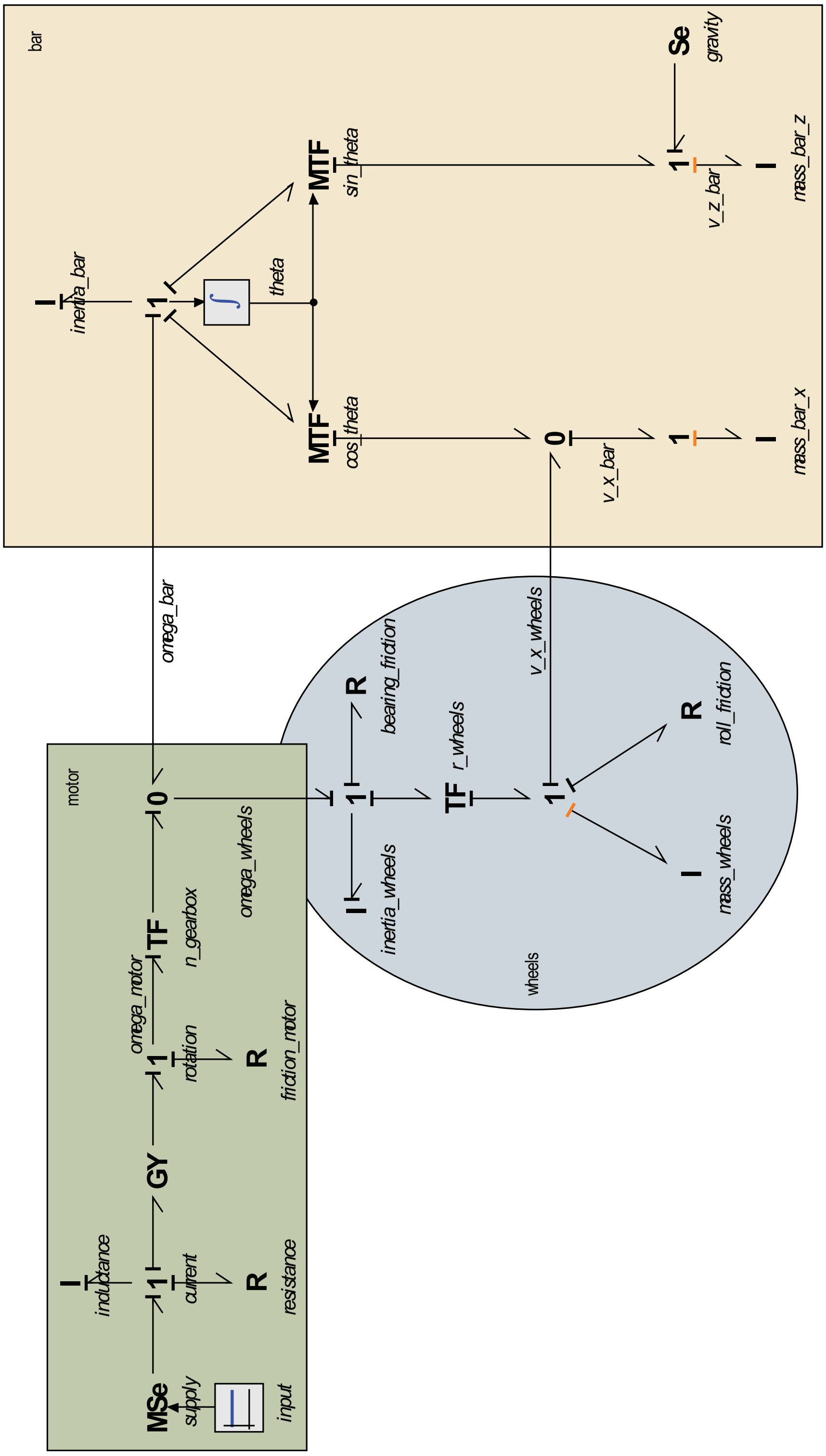
motor
I
inductance
MSe
supply
input
1
current
R
resistance
GY
1
rotation
R
friction_motor
omega_motor
TF
n_gearbox
0
omega_wheels
omega_bar
bar
I
inertia_bar
1
theta
MTF
cos_theta
MTF
sin_theta
0
v_x_bar
1
I
mass_bar_x
1
v_z_bar
I
mass_bar_z
Se
gravity
wheels
I
inertia_wheels
1
R
bearing_friction
TF
r_wheels
1
v_x_wheels
I
mass_wheels
R
roll_friction

Figure 2.2: Bond graph model of the physical system

The inertia of the bar is described in the rotation domain. It was modeled as the inertia of a rod rotating around one of its ends, as this was the most obvious approximation to the expected mechanical design.

$$I_{bar} = \frac{1}{3} m_{bar} \cdot l_{bar}^2 \tag{2.4}$$

An I-type storage element connected to the one-junction representing the rotation, represents the inertia.

The mass of the bar is described in the translation domain. Two identical I-type storage elements were used; one for the x-direction and one for the z-direction. Gravity 'g' only works vertically and is thus connected to act upon the mass in the z-direction.

A simulation of the bar showed a rotation if a small disturbance from the upright position was given, as was expected. The floor was not implemented in the model, because it is not within the working area of 0.35 rad, and thus the bar moves through the floor. The stable position in the simulation is when the model is up side down.

Wheels

This submodel also has the rotation and translation domain. The flow variables are the angular velocity ω_{wheels} and the translational velocity v_{wheels} respectively, which are related by:

$$v_{wheels} = \omega_{wheels} \cdot r_{wheels} \tag{2.5}$$

This transformation is done by a TF-element.

The wheels, as well as a possible gear mounted to the axis of the wheels, have an inertia, which is described in the rotation domain. It was expected that a wheel with a rubber tire has more mass to the outside than to the inside. Obviously, the gear has a smaller radius than the wheels. The combination of these two inertia's was approximated by the inertia of a solid cylinder.

$$I_{wheels} = \frac{1}{2} m_{wheels} \cdot r_{wheels}^2 \tag{2.6}$$

The axis is subject to bearing friction, which is described in the rotation domain as well.

In the translation domain, the mass of the wheels is present. Also, roll friction due to small deformations of the wheels while they are in contact with the floor, was implemented. According to the specifications, the system has to function on a non-slippery floor. Therefore, it was assumed that the wheels do not slip and thus this was not implemented in the model.

A simulation of the wheels in which a constant positive torque to the axis is applied, showed a movement in the positive direction, as expected.

Motor

The standard bond graph model of an electric motor was used, which has the electrical and the mechanical (rotation) domain. A TF-element was added to the rotation domain, to model a lossless gearbox to increase the torque delivered by the motor. An I-type storage element in the rotation domain, modeling the inertia of the motor axis and a possible small spur gear mounted to it, was neglected. This inertia is significantly smaller than the inertia of two wheels and a larger spur gear (for increasing torque), both with much larger diameters.

A simulation of the motor basically shows the same behavior as the wheels. While a constant positive voltage is applied, the motor rotates in the positive direction.

2.3 Plant Simulations

The model of the plant was simulated in 20-Sim to verify its behavior. Two simulations are shown, both with an initial angle $\theta = 0$ rad. First, a disturbance given to the bar is simulated. Second, the situation where a constant positive voltage is applied to the motor is simulated.

Figure 2.3 shows the first simulation, in which a disturbance is given.

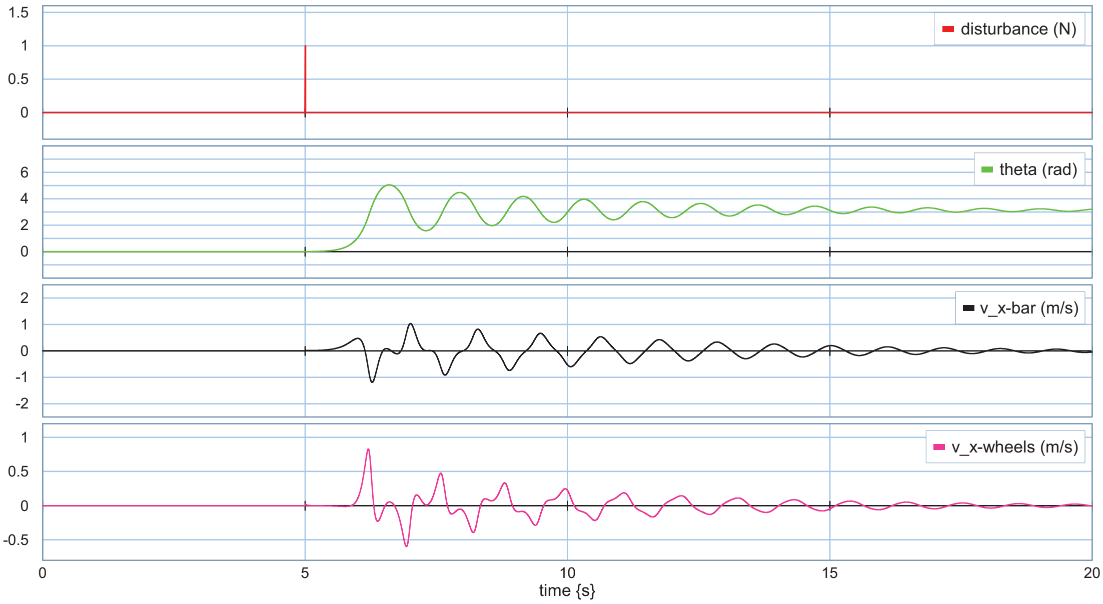

Figure 2.3: Simulation of the plant when a disturbance is given

For five seconds, the model stands upright in a meta-stable position. At t = 5 s, a disturbance is given to the bar in the positive x-direction. The model falls in the positive direction and as described in Section 2.2.2, it is not limited by a floor. It oscillates around the upside down position (angle of π rad) and slowly comes to a stop. The wheels are oscillating as well, due to the rotating bar. This simulation showed what was expected.

Figure 2.4 shows the second simulation, in which a constant positive voltage is applied to the motor. Again, for five seconds, the model stands upright in its meta-stable position. At t = 5 s, the motor is powered continuously to give a positive rotation to the wheels. Because of this, the bar falls in the negative direction and oscillates around the upside down position again (angle of $-\pi$ rad). The velocity of the wheels is influenced by the rotating bar and by the motor. Eventually, both the bar and the wheels have a certain constant positive velocity due to the continuously powered motor. Again, this simulation showed what was expected.

2.4 Conclusion

A bond graph model of the physical system was made using a planar mechanics description and simulations were done to verify the model. Because the simulation results agreed with expectations, the model was assumed to be plausible.

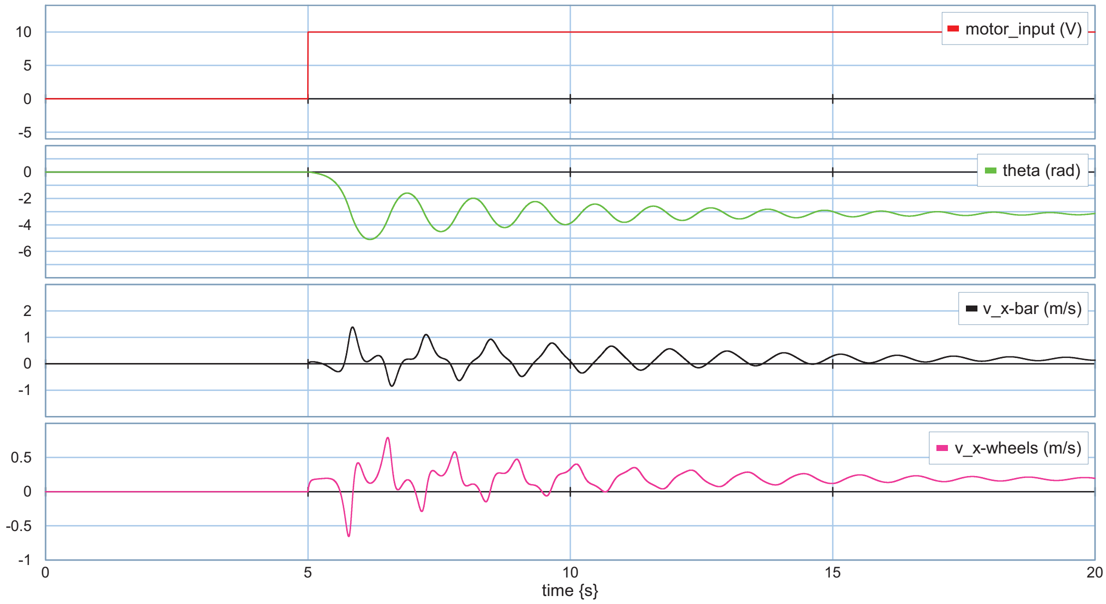

Figure 2.4: Simulation of the plant when the motor is powered

3

Controller Design

The system is intrinsically unstable, so it falls over when the center of mass of the bar is not positioned exactly above the axis of the wheels, as shown in simulations in Section 2.3. A continuous controller was designed to keep the system upright, as it is designed more easily than a discrete controller because it is not influenced by e.g. a sample frequency. The designed controller was tested in simulations using the model of the plant and was then discretized and tested again.

Section 3.1 describes a controller criterion and specifications. In Section 3.2, the design and testing of the continuous controller is presented. This controller is discretized and tested, which is described in Section 3.3.

3.1 Criterion and Specifications

The controller is designed using the pole-placement method in the complex s-plane. With this method, poles and zeros are placed in the s-plane, so that the dominant poles of the closed loop system are placed at a desired location:

$$s = -\alpha \pm j\beta \tag{3.1}$$

To design a controller, a criterion is needed. For this system, the angle θ (see Figure 2.1) is controlled to zero. The specifications that were used while designing the controller, are shown in Table 3.1.

Table 3.1: Controller specifications

Peak-time (t_{peak})	0.5 s
Relative damping (z)	$\frac{1}{2}\sqrt{2}$

The peak-time of the response to a step on the input is defined as:

$$t_{peak} = \frac{\pi}{\beta} \tag{3.2}$$

For this system, the peak-time means the time that is needed to reach the first overshoot peak after compensating an impulse disturbance. The peak-time specification was derived from simulations of a self-balancing scooter made by van Kuppeveld [5]. As a first specification for the correct order of magnitude, a peak-time of $t_{peak} = 0.5$ seconds was chosen.

The relative damping determines the settling behavior of the step response, e.g. the amount of overshoot. An underdamped system results in oscillations and an overdamped system results in an unstable system as the output of the controller can not keep up with a fall. Thus, a slightly underdamped system was preferred. A relative damping of $z = cos(\frac{1}{4}\pi) = \frac{1}{2}\sqrt{2}$, meaning 4 percent overshoot, was used as a design specification.

From this, the location of the dominant complex poles was found.

$$\beta = \frac{\pi}{t_{peak}} = \frac{\pi}{0.5} = 2\pi \tag{3.3}$$

Because $z = \frac{1}{2}\sqrt{2}$, α and β in (3.1) are equal. This yields for the location of the dominant complex poles of the closed loop system:

$$s = -2\pi \pm 2\pi j \tag{3.4}$$

3.2 Continuous Controller

A block diagram of the system with feedback on the angle of the bar is shown in Figure 3.1.

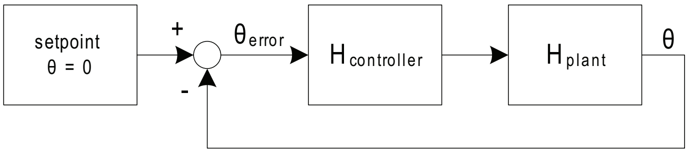

Figure 3.1: A standard block diagram of a controlled system

The setpoint of the angle of the bar is set to zero, according to the criterion. The plant and the controller are represented by transfer functions, which indicate the location of the poles and zeros. A transfer function from an input to an output of the plant is found by linearizing the non-linear model (Section 2.2.2) around an interesting point. As the robot needs to be balanced, the model is linearized around $\theta = 0$ rad. The input of the model is the signal to the motor and the output is the angle θ of the bar. The linearization was done by 20-Sim, which outputs the transfer function (3.5).

$$H_{plant}(s) = \frac{-2.287s - 1.138}{s^3 + 38.06s^2 - 44.72s - 876.8} \tag{3.5}$$

When the transfer function of the controller $H_{controller}(s)$ equals 1, a root-locus of a unity feedback system is found, indicating the possible locations of the poles of the system and thus indicating its behavior and stability. Figure 3.2 shows the root-locus for positive variations of the root-locus gain K'.

It can be seen that there is a pole in the right half of the s-plane, resulting in an unstable system. Obviously, a simple proportional controller can not stabilize the process, as there will always be a pole in the right half plane when increasing the root-locus gain. This pole will only be in the left half plane for very high root-locus gains, but then the other two poles have large imaginary parts which results in severe oscillatory behavior. Canceling the pole by a zero is in practice not possible, because every slight deviation from the location of the pole results in the system revealing unstable behavior. The part of the root-locus of the unstable pole was bent to the left half plane, in order to get a stable system. To achieve this, a pole must be in the left half plane to the right of the zero close to the origin.

The PID-controller of (3.6) was designed by pole/zero-placement.

$$H_{controller}(s) = K'_{controller} \cdot \frac{(s+23)(s+8)}{s(s+80)} \tag{3.6}$$

The proportional factor ensures a larger controller output when the angle gets larger, the differentiating factor ensures a larger output when the rate of change of the angle is large and the

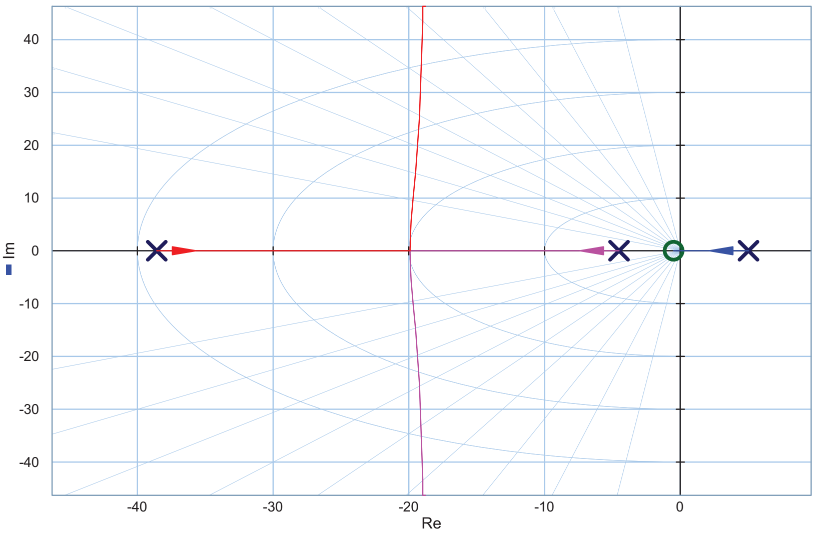

Figure 3.2: Root-locus of the plant with unity feedback

integrating factor compensates a constant disturbance at the output. Clearly, this controller has zeros in $s = -8$ and -23 and poles in $s = 0$ and -80. A pole/zero-plot showed that for a root-locus gain of $K' = 1848$, the dominant complex poles are placed in $s = -2\pi \pm 2\pi j$, according to (3.4). Figure 3.3 shows the desired location of the poles and zeros of the closed loop system. All poles are located in the left half plane, thus the system is stable.

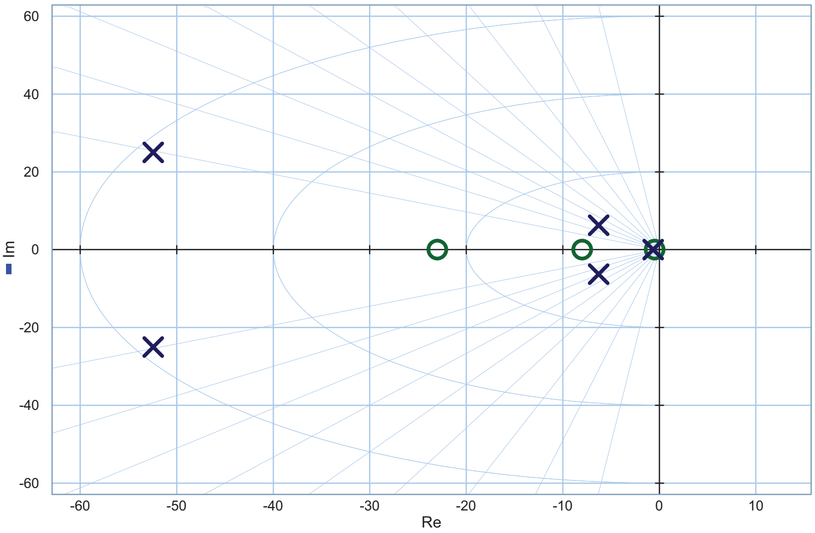

Figure 3.3: Desired location of the poles of the closed system

The proportional factor and time-constants were found by rewriting (3.6) in the standard series representation of a PID-controller, (3.7), yielding (3.8).

$$H_{controller}(s) = \frac{K_p}{\tau_i} \cdot \left(\frac{\tau_i s + 1}{s}\right) \cdot \left(\frac{\tau_d s + 1}{\tau_d \beta s + 1}\right) \tag{3.7}$$

When substituting the tameness constant $\beta = 0.1$ (the pole is a factor 10 left of the zero), (3.7) becomes:

$$H_{controller}(s) = \frac{K_p}{\frac{1}{23}} \cdot \left(\frac{\frac{1}{23}s + 1}{s} \right) \cdot \left(\frac{\frac{1}{8}s + 1}{\frac{1}{80}s + 1} \right) \tag{3.8}$$

Hence, $\tau_d = \frac{1}{8}$ and $\tau_i = \frac{1}{23}$. The factor K_p was calculated using the time constants.

$$K'_{controller} = \frac{K_p}{\tau_i} \cdot \frac{\tau_{z_1} \cdot \tau_{z_2} \cdot \tau_{z_3} \cdot \ldots}{\tau_{p_1} \cdot \tau_{p_2} \cdot \tau_{p_3} \cdot \ldots} = \frac{K_p}{\frac{1}{23}} \cdot \frac{\frac{1}{23} \cdot \frac{1}{8}}{\frac{1}{80}} = 1848 \tag{3.9}$$

Thus:

$$K_p = K'_{controller} \cdot \frac{1}{10} = 1848 \cdot \frac{1}{10} = 184.8 \tag{3.10}$$

At first, the controller was tested in a feedback loop using the linear model (see Figure 3.1). A simulation showed that the angle θ was controlled to zero smoothly. The peak-time after a disturbance was less than 0.5 seconds.

A simulation with the non-linear model of the plant was done as well. See Figure 3.4 for the simulation result. A disturbance was given to the bar at $t = 1$ s. The system does not fall over and the angle θ of the bar is controlled to zero smoothly. The peak-time after the disturbance is about 0.6 seconds. This simulation shows that the controller settings also apply to the non-linear model.

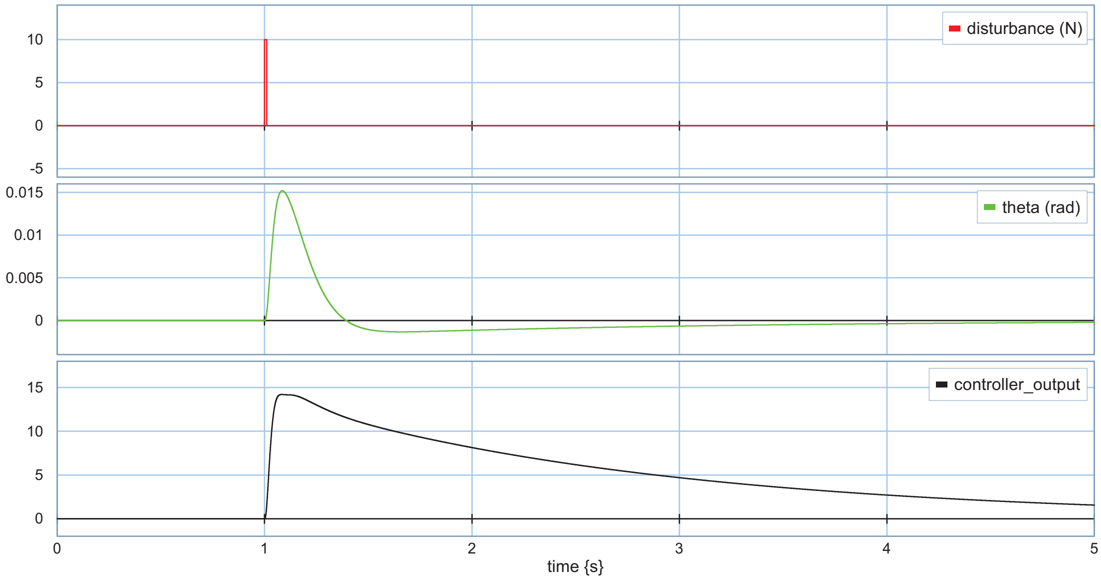

Figure 3.4: Continuous controller simulation using non-linear plant model

3.3 Discretizing

The controller is realized digitally, so the designed continuous controller was discretized using the bilinear transform. This transformation maps every point of the left half of the s-plane to the interior of the unit circle in the z-plane, which means stability is preserved. The bilinear transform is in fact a substitution of the Laplace variable s, as denoted in (3.11).

$$s \leftarrow \frac{2}{T_s}\frac{z-1}{z+1} \tag{3.11}$$

Substituting (3.11) into (3.7) and reworking yields:

$$H_{controller}(z) = \frac{K_p}{\tau_i} \cdot \frac{(2\tau_i + T_s)z - 2\tau_i + T_s}{2(z-1)} \cdot \frac{(2\tau_d + T_s)z - 2\tau_d + T_s}{(2\tau_d\beta + T_s)z - 2\tau_d\beta + T_s} \tag{3.12}$$

T_s denotes the sample time of the controller, which was chosen to be 4 ms. From this, the new parameters were calculated, yielding $K_p = 184.8$, $\tau_d = 0.254$, $\tau_i = 0.091$ and $\beta = 0.1$.

To test the discrete controller, the model of Figure 3.5 was used.

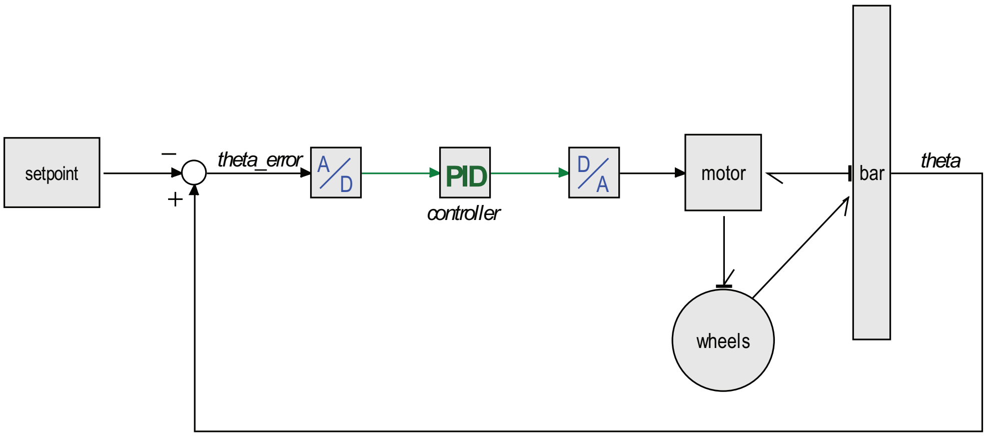

Figure 3.5: Model for testing the discrete controller

The range of the 10-bits A/D and D/A-converter was set to ± $\frac{1}{2}\pi$ (rotation of the model to the floor in both directions) and ± 50 (derived from the continuous simulation) respectively. The simulation result is shown in Figure 3.6. It can be seen that the performance of this discrete controller is worse than the performance of the continuous controller. Although the system does not fall, the angle θ is not controlled to zero very well, which was found to be mainly due to the accuracy of the A/D-converter. Also, the peak-time after a disturbance has increased to about 1.3 seconds, which was found to be mainly due to the conversion from a continuous to a discrete controller. However, the simulation does show a controller which stabilizes the plant. At this time, the controller was not refined any further, because some parts of the model were still in ideal form, e.g. the sensors.

3.4 Conclusion

A continuous controller, controlling the angle θ of the bar to zero, was designed according to specifications. Simulations showed an operational continuous controller. From this controller, a discrete controller was realized which was able to stabilize the intrinsically unstable process, although the performance was worse than the performance of the continuous controller. A refined controller is described in Chapter 4, where the ideal parts of the model are replaced by non-ideal parts.

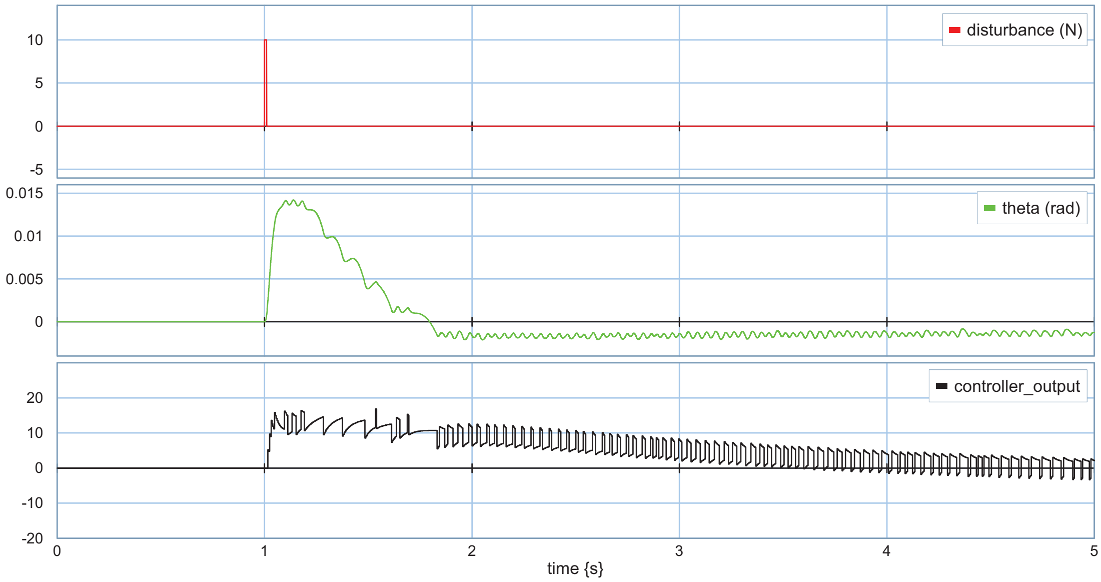

Figure 3.6: Discrete controller simulation

4

System Design and Realization

In this chapter, an embedded system design is described, as well as the electronical and mechanical designs of the system. Section 4.5 presents a simulation of the final system model.

The system was designed according to Figure 4.1, which gives a block diagram of the complete realized system. In this figure, 'y' denotes an arbitrary controller output, 'DC' a duty-cycle, 'T' a torque, 'V' a voltage and '[0..2^{10}-1]' a 10-bits digital value.

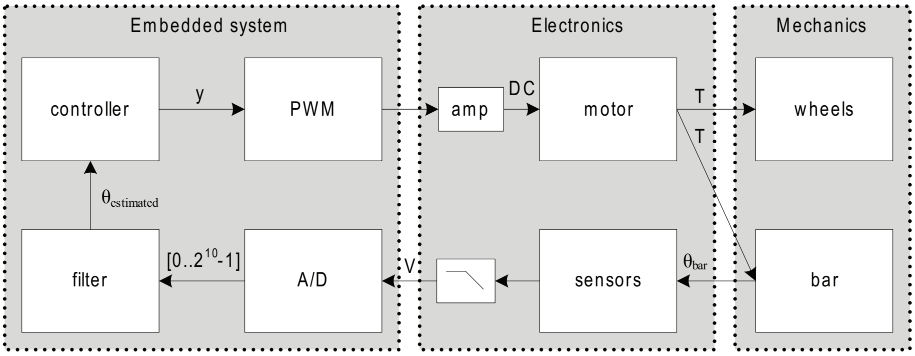

Figure 4.1: Block diagram of the realized system

From this figure it is shown that the system is divided in three different disciplines, i.e. the embedded system, electronical and mechanical discipline.

4.1 Embedded System Design

The necessary signal processing was done digitally, because of the relatively easy implementation and flexibility of embedded software. A microcontroller was used to do this and from the different possible processors given in Table 4.1, the Atmel AtMega32 was chosen because of its high performance, low power consumption, low cost and a readily available small printed circuit board (PCB), i.e. the MiniMegaBoard designed by Edwin Dertien [4].

As shown in Figure 4.1, the embedded system takes care of the A/D-conversion of sensor signals, filter calculations out of the digitized measurements, control action calculation on the angle estimate and a pulse width modulation (PWM) duty-cycle calculation from the controller output. Also, some diagnostic functions were added, i.e. LED's to indicate a positive or negative bar angle, the directional dominance of the motor and the operational 'heart beat' of the microcontroller. These functions are performed inside a loop (the main loop), which is run forever. As no 20-

Table 4.1: Different possibilities for the embedded hardware

Name	ARM TS7400	Atmel AtMega32	Microchip PIC18F4525
Type	32 bits	8 bits	8 bits
Speed	200 MHz	16 MHz	40 MHz
MIPS	≈ 133	16	10
Supply current at 5 V	350 mA	22 mA	23 mA
Price	≈ €100	≈ €10	≈ €7
I/O	20	32	36
PWM-outputs	no	4 channels	4 channels, full H-bridge
A/D-channels	4 channels, 12 bits	8 channels, 10 bits	13 channels, 10 bits
Floating point	yes	no	no
Flash, SRAM, EEPROM	32 MB, 32 MB, -	32 kB, 2 kB, 1 kB	24 kB, 4 kB, 1 kB

Sim code-generation library was available for the Atmel microcontroller, the software was written manually.

4.1.1 Peripheral usage

The AtMega32 has an 8 channel 10-bits A/D-converter onboard, which was set up to convert signals from 0 V to the supply voltage V_{dd} (5 V), where an output of 0 denotes 0 V and an output of 1023 denotes V_{dd}. Thus, the least significant bit (LSb) represents $\frac{5}{1024} = 4.88$ mV at the input of the A/D-converter.

The onboard 4-channel PWM-module was used to output a duty-cycle to drive the motor, as this is an easy way of controlling its direction. A PWM-frequency of 50 Hz was used (20 ms period), as the mechanical time-constant of the motor is 19 ms. The module outputs to two pins of the microcontroller in a complementary way, which means that one pin always is the complementary of the other (see Figure 4.2).

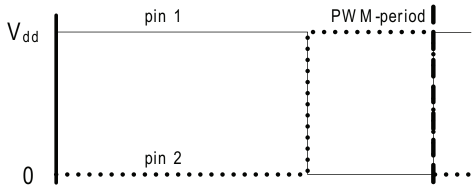

Figure 4.2: PWM-outputs configuration showing a pin1 duty-cycle of ≈ 66 %

To these pins, the motor is connected via an amplifier. By changing the duty-cycle, the motor can be rotated in both directions. A duty-cycle of 50 % means there is no dominance in rotation of the motor, as the motor rotates the same amount of time in the positive and negative direction. Otherwise, a dominance exists in the positive or negative direction when the duty-cycle is greater than or less than 50 % respectively. The PWM-module uses a timer of the microcontroller for period and duty-cycle timings.

The main-loop needs to be run at a constant frequency, e.g. to ensure correct differential value calculations. A timer (TMR2) was used to do this. This timer starts counting at the beginning of the main-loop. When the loop is finished executing, the program waits for a TMR2 count overflow to occur which corresponds to a predefined amount of time and then starts the main loop again. This way, the duration of the loop is always the same, independent of calculation time due to arbitrary values of variables. Figure 4.3 shows a timing diagram, in which 'o' stands for overflow.

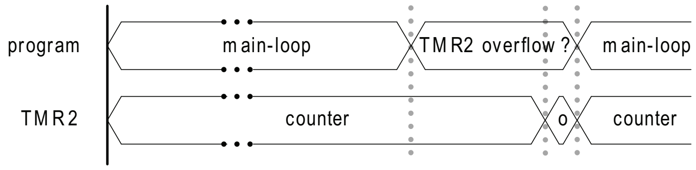

Figure 4.3: Timing diagram of the software

4.1.2 Implementation

AVR-studio 4 was used to write a program in C, which was compiled by the AVR-GCC-compiler. The AtMega32 was programmed by AVR-studio using an AVR-ISP (Incircuit Serial Programmer), modified by Edwin Dertien to fit the MiniMegaBoard.

The output of the A/D-converter is the input to the filter, which calculates the angle the bar makes, according to (4.5). The PID-controller was implemented by inserting equations from the discrete 20-Sim PID-submodel. A PWM-duty-cycle was calculated from the controller output. A positive controller output results in a duty-cycle greater than 50 %, a negative output results in less than 50 %. The larger the absolute value of the controller output, the larger the absolute value of the duty-cycle. Of course, the duty-cycle has a minimum and maximum of 0 % and 100 % respectively, which means that one pin is always on and the other is always off.

Calculations are mostly done using floating-point numbers and according to multiple simulations inside AVR-studio, the maximum duration of running the complete loop once was about 3.7 ms. Because of this, TMR2 was set to count to 4 ms before generating an overflow, so Figure 4.3 always holds. Thus, a sample frequency of 250 Hz was achieved. Care was taken not to exceed the Nyquist frequency of 125 Hz at the input of the A/D-converter, by means of anti-alias filters.

Figure B.1 shows a flow diagram of the code which is executed inside the microcontroller.

4.2 Angle Measurement

To be able to control the angle of the bar by means of the PID-controller, the angle was measured by two sensors.

4.2.1 Sensors

From the possible sensors shown in Table 4.2, the angular rate and acceleration sensors were used to measure the angle because of their good response time and because they are not influenced by the terrain.

Table 4.2: Properties of angle measuring sensors

Sensor type	Optical (IR)	Acoustical	Tilt	Angular rate	Acceleration
Measurement	Distance	Distance	Absolute tilt	Angular velocity	Acceleration
Range	≈ 0 - 0.3 m[1]	≈ 0 - 1 m[1]	max. $\approx \frac{1}{2}\pi$ rad	max. $\approx 2\pi$ rad/s	max. ≈ 2450 m/s^2
Price	€10	€10	€150	€75	€10
Drift free angle?	yes	yes	yes	no	yes
Response time	++	+	−−	++	+
Influenced by . . .					
. . . velocity?	no	no	no	no	yes
. . . placement?	yes	yes	no	no	yes
. . . terrain?	yes	yes	no	no	no

---

[1] This depends on the measurement method. The given values are estimates when using 5 V and 100 mA.

A gyroscope is an angular rate sensor. It measures the angular velocity ω which is integrated to an angle θ. Because this is a noisy measurement, the integrated signal suffers from drift, which is compensated by means of a filter (Section 4.2.2). From simulations, it could be seen that the maximum angular velocity of the system is about 3 rad/s. Thus, the Analog Devices ADXRS300EB (Evaluation Board) was chosen, which is capable of measuring $5\ ^1/_4$ rad/s. A low-pass filter is integrated on-board, with a cut-off frequency of $f_{\text{cut-off}} = 40$ Hz. This filter also acts as an anti-alias filter.

To compensate the drift effect, a single axis accelerometer was used which is placed in such a way that it measures the acceleration due to a change in the angle θ (see Figure 4.4). The accelerometer measures a component of the gravitational and translational acceleration, i.e.:

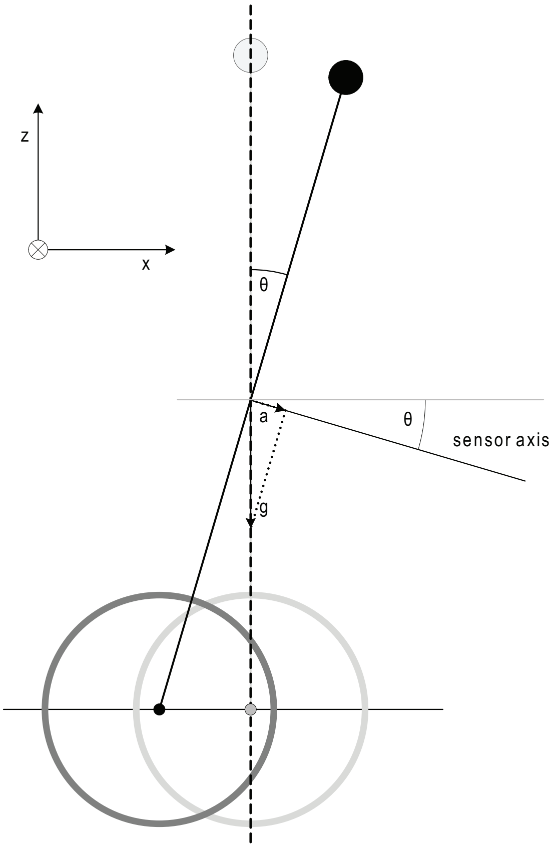

Figure 4.4: Accelerometer placement on the robot

$$a_{sensor}(t) = g \cdot sin\,\theta + a_x \cdot cos\,\theta \tag{4.1}$$

When the sensor is placed in the center of mass of the system, the *translational* acceleration due to a change in the angle is minimized (ideally zero). This is because the center of mass is defined as the point around which a body will continue to rotate in the absence of external forces and torques, after some disturbance has caused the rotation [1]. Moreover, the translational movement was expected to be very small and thus the translational acceleration is close to zero. Hence, the second term was neglected.

Furthermore, because $sin\,\theta = \theta$ for small deviations around $\theta = 0$ rad, (4.1) reduces to:

$$a_{sensor}(t) = g \cdot \theta \tag{4.2}$$

Thus, an angle was found directly from the acceleration measured by the accelerometer.

$$\theta(t) = \frac{a_{sensor}}{g} \tag{4.3}$$

When using this linearization, the error made at 0.175 rad (maximum angle according to specifications) $\approx$ 0.78 %, which is acceptable.

The maximum acceleration measured by the sensor is found at an angle of 0.175 rad.

$$a_{max} = g \cdot \theta = 9.81 \cdot 0.175 \approx 1.72\ \mathrm{m/s^2} \tag{4.4}$$

The Freescale Semiconductor MMA1260EG accelerometer was chosen, which can measure 1.5 g (14.7 m/s^2). The output of the sensor was connected to a low-pass filter, because only the low frequency variations, i.e. the steady-state position, are interesting. Again, this filter also acts as an anti-alias filter. The cut-off frequency is $\mathrm{f_{cut\text{-}off}}$ = 3.4 Hz.

Both sensors were placed in the center of gravity of the system, which is the middle of the bar.

4.2.2 Drift compensation filter

The angular velocity ω measured by the gyroscope is integrated to an angle θ. Because noise is always present, the integrated signal suffers from drift. Therefore, a filter was used to compensate the drift and which provides a good angle estimation out of the measurements of the gyroscope and the accelerometer.

A blockdiagram of the implemented filter is shown in Figure 4.5.

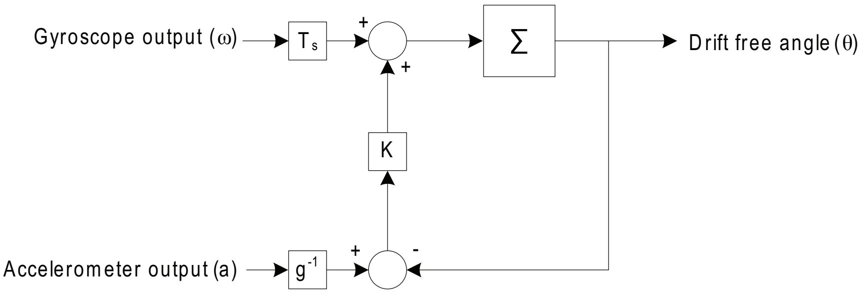

Figure 4.5: Drift compensation filter

The equation for this filter is given by:

$$\theta = \theta_{previous} + (\omega \cdot T_s + K \cdot (\frac{a}{g} - \theta_{previous})) \tag{4.5}$$

T_s denotes the sample time. The angle difference between two sample moments measured by the gyroscope is $\omega \cdot T_s = \Delta\,\theta_{gyroscope}$. K determines the amount that is added to $\theta_{gyroscope}$, the angle measured by the gyroscope, to compensate drift.

The filter functions as follows. Assume all initial values are zero, T_s and K are one and the system is in the upright position. The gyroscope measurement is noisy, so when there is no angular rotation, as is the case in the upright position, it measures a certain amount, say $\omega = 0.1$ rad/s. When no filter was applied, integrating (or summing in the discrete domain) such a step function would result in a ramp function. Now, however, $\Delta\,\theta_{gyroscope} = 0.1$ rad. The accelerometer output ideally equals zero, because the system is in the upright position. Thus, the drift free angle is $\theta = 0.1$ rad. At the next sample instant, $\Delta\,\theta_{gyroscope}$ still is 0.1 and the accelerometer output equals zero. However, the output of K now is -0.1, so the input to the summator becomes: $0.1 + -0.1 = 0$. Thus, the drift-free angle still equals 0.1. The drift effect is compensated, but the filter output has a certain offset.

In reality, the accelerometer has some high-frequency noise present on the output signal. By choosing K small, only the low-frequency component (steady-state position) is passed through, as this component gets the time to build up after many runs through the filter loop, and the noise is canceled out.

Changing T_s and K will influence the offset value. A sampletime of $T_s = 4$ ms (as described in Section 4.1) and $K = 0.02$ was chosen, which resulted in an offset value of 0.008 rad according to simulations. This offset was subtracted from the filter output.

4.2.3 Performance

Both sensors were connected to an A/D-channel on the microcontroller. This 10-bits converter reduces the accuracy of the measured angle. As noted earlier, the A/D-converter was set up to convert from 0 V to 5 V, so its sensitivity equals $\frac{5}{1024} = 4.88$ mV/bit. For the accelerometer, the following holds:

$$\text{resolution}_{\text{accelerometer}} = 1200\ \text{mV/g} = 122.3\ \text{mV/m/s}^2 \tag{4.6}$$

So, the LSb corresponds to $\frac{4.88}{122.3} = 0.04$ m/s^2 at the input of the A/D-converter. This translates to an angle sensitivity of the A/D-converter according to (4.3) by: $\frac{0.04}{9.81} = 0.0041$ rad/bit.
For the gyroscope, the following holds:

$$\text{resolution}_{\text{gyroscope}} = 286.5\ \text{mV/rad/s} \tag{4.7}$$

So, the LSb corresponds to $\frac{4.88}{286.5} = 0.017$ rad/s at the input of the A/D-converter. As simulations showed that an accelerometer sensitivity of 0.0175 rad/bit and a gyroscope sensitivity of 0.0175 rad/s/bit still allow proper angle control, the accuracy reduction by the A/D-converter caused no problems.

The angle measurement was modeled in 20-Sim, as shown in Figure 4.6.

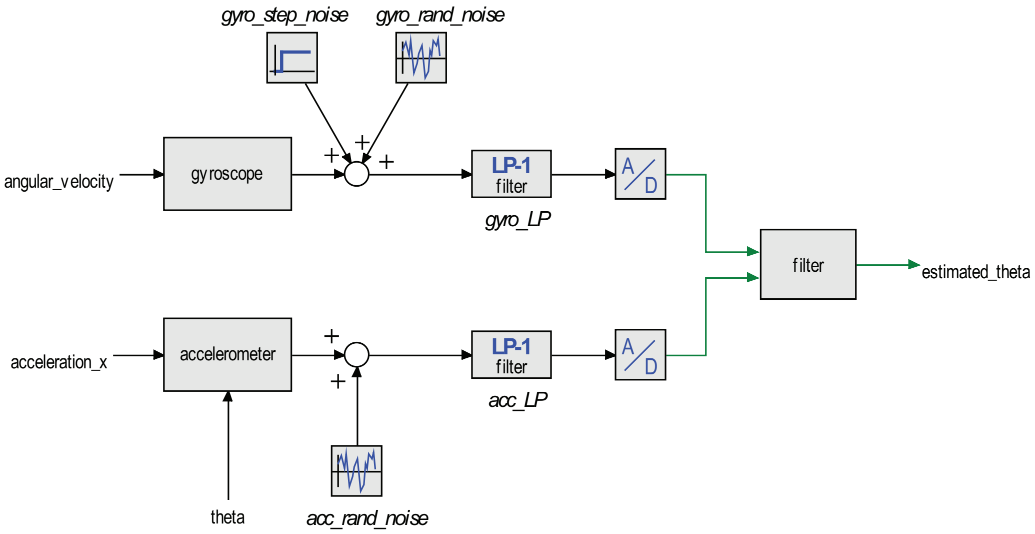

Figure 4.6: Sensors and filter model implemented in 20-Sim

Both sensors were modeled and random noise was added to model non-ideal measurements (taken from the datasheet). A step-function was added to the output of the gyroscope as well, to model the noise which causes the drift effect after integration. The measurements are low-pass filtered and are digitized by the 10-bits A/D-converters. The ranges of the converters were set to the sensor ranges. The drift compensation filter was implemented to provide the angle estimation.

A simulation of a controlled system is shown in Figure 4.7, which shows the actual angle (theta) and the simulated filter output (estimated_theta). A start-up situation is present, but overall the measured angle is drift free and quite robust.

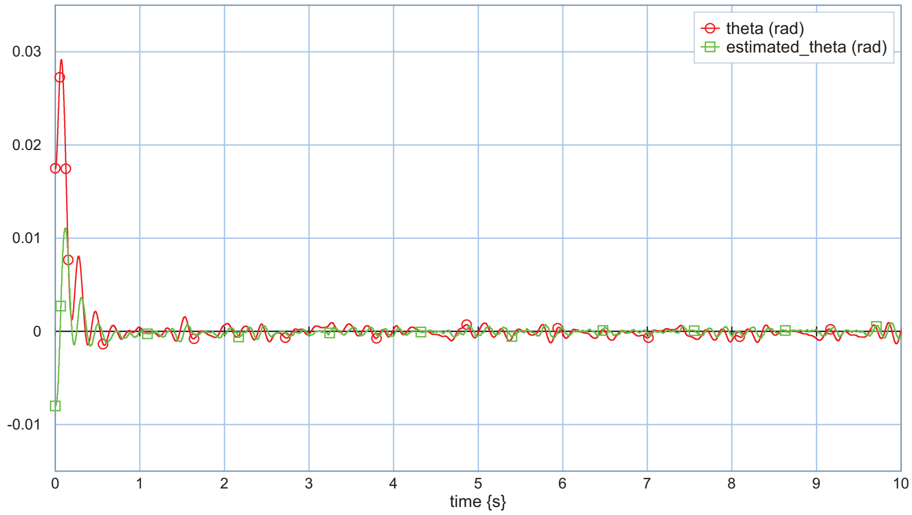

Figure 4.7: Angle measurement simulation on a controlled system with initial angle $\theta = 0.0175$ rad

The drift compensation performance was measured as well, by keeping the system at a constant $\theta \approx 0$ rad and watching a change in the measured angle by means of an LED. For a period of approximately 80 minutes, the measured angle did not change or drift. Thus, the filter was assumed to be suitable.

4.3 Electronics

The electronic components of the system, besides the microcontroller (see Section 4.1), are the power supply, the motor with its amplifier, sensor interfacing and diagnostic LED's. Electronic schematics are given in Figure B.2.

4.3.1 Power supply

The system needed to be powered by standard batteries, according to specifications. Thus, six 1.5 V 2700 mAh AA-batteries were used, which add up to 9 V. The motor is directly fed from this 9 V supply. However, the microcontroller and sensors use a supply voltage V_{dd} of 5 V. This is achieved by connecting a 5 V regulator IC to the 9 V battery voltage, i.e. the National Semiconductor LM2937 in SOT223-package. 9 V is then converted to 5 V, where the remaining 4 V is dissipated as heat. The load current of the IC is more than 500 mA.

Also, a 9 V power supply can be plugged into the system to relieve the batteries. When batteries run low, their voltage drops which influences the measurements and overall performance of the system. An external power supply ensures 9 V is applied all the time.

A large capacitor of 4700 μF was placed across the power supply for buffering. When the motor starts up, it draws a lot of current (2700 mA) for a very short period of time. When no buffer is used, the battery voltage would drop because of the peak-current. This would cause a drop of the supply voltage to the microcontroller and sensors, which would result in a reset of the microcontroller.

4.3.2 Motor and motor driver

The maximum torque which needed to be delivered by the motor was found from simulations. An effort sensor was placed in the model of the motor right in front of the zero-junction of Figure 2.2. The system was left uncontrolled, like the simulation of Figure 2.3. At an angle of 0.175 rad (maximum angle according to specifications), a torque of $\approx$ 12 mNm was found to be delivered. A 5 W Maxon 110119 motor was used which operates at a nominal voltage of 9 V. This motor can deliver a continuous torque of 6.43 mNm at a continuous current of 751 mA. As 12 mNm should be delivered, a gear ratio of at least $\frac{12}{6.43} = 1.87$ was needed to increase the delivered torque by the motor. Two spur gears (40 and 120 teeth) were used to create a gear ratio of 3.

The motor can not be driven directly from the PWM-outputs of the microcontroller, so an amplifier was needed. An H-bridge-IC was used to amplify the outputs, i.e. the Vishay Si9986. The supply voltage of the H-bridge is the battery voltage, i.e. 9 V. The PWM-outputs of the microcontroller were directly connected to the H-bridge, as a FET-driver was integrated. Two H-bridges were placed in parallel due to an error in the datasheet, which stated that the IC could drive a continuous current of 1 A at 12 V while it can only achieve 1 A at 5 V. As at least 751 mA at 9 V was needed, one H-bridge was not sufficient.

4.3.3 Additional electronics

The sensor interfacing comprises of two low-pass filters, which was mentioned in Section 4.2.1. The gyroscope filter was integrated on the sensor board, the accelerometer filter was a simple first order low-pass filter using a 47 μF capacitor and a 1 kΩ resistor.

In total, five diagnostic LED's were used to indicate the behavior of the system. Figure 4.8 shows an overview.

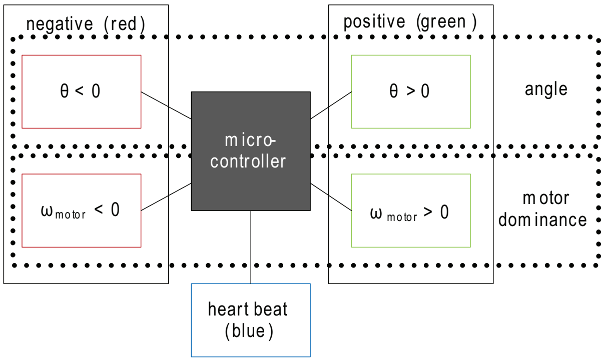

Figure 4.8: Overview of the diagnostic LED's

Two were used to indicate a positive or negative bar angle (green for a positive and red for a negative angle), two were used to indicate rotational dominance of the motor (green for positive and red for negative dominance) and one LED (blue) was used to show the microcontroller still runs the main loop. All LED's were connected in such a way that current was sunk by the microcontroller instead of sourced, as a larger current can be sunk than sourced.

4.4 Mechanical Construction

The mechanical construction needed to be rigid and robust. It mainly consists of an aluminium frame, through which an axis can freely rotate by means of ball bearings, and two wheels connected to both ends of that axis. The wheels are made of plastic with rubber tires around them, to provide grip. The axis length was chosen relatively large, to justify the neglection of sideward movements.

A gear was mounted to the axis, to increase the torque of the motor. The frame was assumed rigid, as the frame could not visibly be deformed. Also, no play in the bearings was seen. Thus, this design is in accordance with the rigid body model presented in Section 2.2.

Figure B.3 shows the hardware construction of the robot, with its measurements. It can be seen that the motor is not mounted in the middle of the robot, but a bit to the back, which results in a constant force pulling the robot in the negative direction. This effect is compensated in the software of the embedded system, by adding a negative offset value to the angle. This way, the motor has a rotational dominance in the negative direction (when standing upright) and is thus compensating the fall backwards. The entire construction, including the connected motor (54 g), the gears (39.1 g) and the batteries (140 g), weighs 529.2 g.

4.5 System Simulation

The final model which resulted after the design steps, is shown in Figure 4.9 (cf. Figure 4.1).

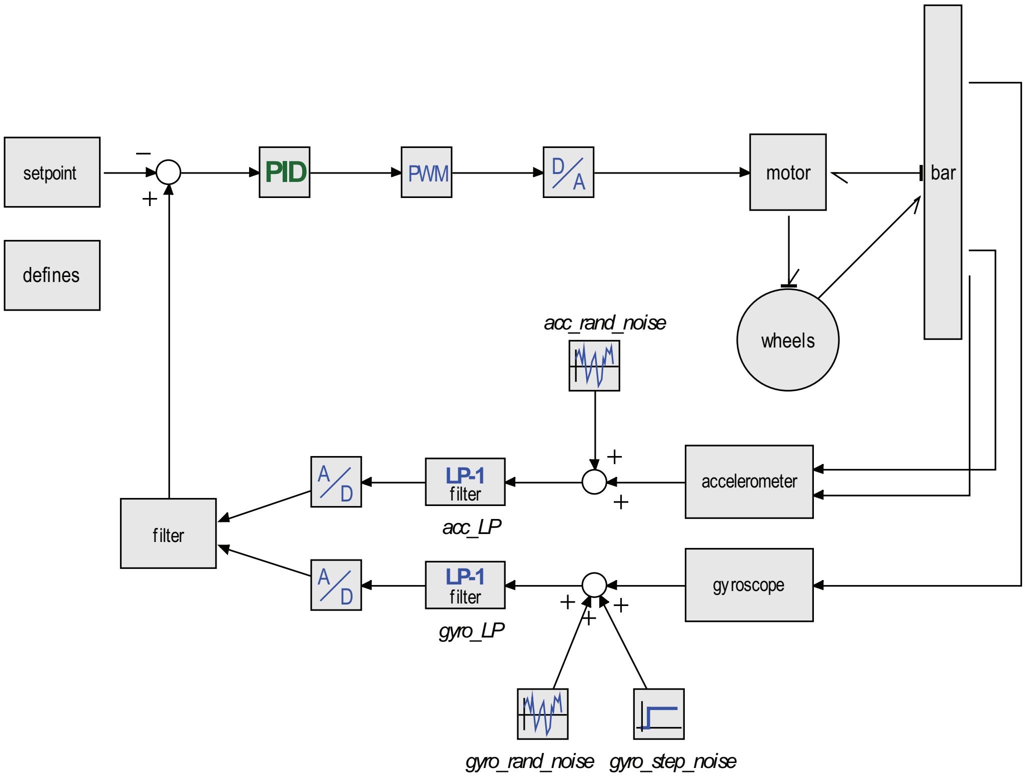

Figure 4.9: Model of the complete system

The output of the discrete controller determines the duty-cycle generated by the PWM-generator. The amplitude is always 9 or -9, which corresponds to the 9 V supply to the motor. A D/A-converter is needed to convert from the digital to the analog domain. The bar of the plant provides the actual angle, the angular velocity and the acceleration in the x-direction for the sensor models. The angle estimated by the filter is used as the feedback variable.

The controller designed in Section 3.3 was not refined further at that time, because many parts of the system were still in ideal form. With this model, the controller was refined by changing its parameters iteratively. The final controller parameters are: $K_p = 300$, $\tau_d = 0.2$ s, $\tau_i = 0.1$ s and $\beta = 0.1$. A simulation of the model of Figure 4.9 with the final parameters is shown in Figure 4.10.

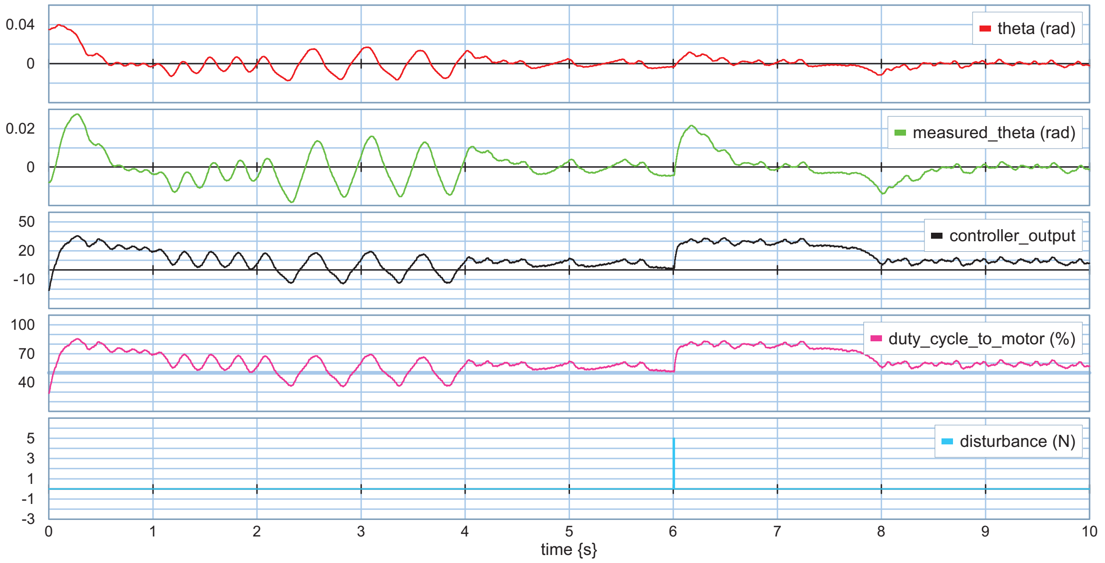

Figure 4.10: Simulation of the system, with initial angle of $\theta = 0.035$ rad

The system is able to compensate an initial angle, as well as a disturbance given to the bar. When balancing, the angle of the bar is always smaller than ±0.0175 rad.

4.6 Total Costs

Table 4.3 lists the components that were used to realize the system. These components add up to the total costs that were made. The hardware needed for programming the microcontroller is not listed, as this was readily available. An estimate is given if the price is unknown.

Table 4.3: Total costs to realize the system

Type	**Name**	**Price/piece**	**Pieces**	**Total price**
Microcontroller	Atmel AtMega32	€ 8.43	1	€ 8.43
PCB	MiniMegaBoard	€ 7.00	1	€ 7.00
Crystal	Rakon LF A161K	€ 1.17	1	€ 1.17
SMD-Adapter	8 pins RE932-01	€ 4.45	1	€ 4.45
SMD-Adapter	16 pins RE932-05	€ 4.45	1	€ 4.45
Gyroscope	A.D. ADXRS300EB	€ 81.98	1	€ 81.98
Accelerometer	F.S. MMA1260EG	€ 13.62	1	€ 13.62
Motor	Maxon 110119	€ 44.05	1	€ 44.05
H-bridge	Vishay Si9986	€ 3.56	2	€ 7.12
Batteries	8 PK 1.5 V AA	€ 8.68	1	€ 8.68
Battery-holder	Keyston 2465	€ 0.93	2	€ 1.86
Capacitor	MCLPR25V478M22X25	€ 1.55	1	€ 1.55
Switch	1MS1T1B5M1QE	€ 0.94	1	€ 0.94
Button	TP32P0080 - APEM	€ 2.95	1	€ 2.95
Mechanical	Wheels, aluminium, etc.	€ 15.00[2]	1	€ 15.00
Additional	LED’s, resistors, etc.	€ 5.00[2]	1	€ 5.00
				€ 208.25

---

[2] These prices are estimates, as the components were left-overs.

It can be seen that the gyroscope comprises almost half of the total price. For the Mechatronics project, it might be advisable to adjust the specifications and design so only an accelerometer can be used, which is less expensive.

4.7 Conclusion

The self-balancing robot was designed and realized. An embedded system takes care of the signal processing, i.e. A/D-conversion and filter, controller and duty-cycle calculations. For the angle measurement of the bar, a gyroscope and an accelerometer were used. A motor is connected to an H-bridge and two gears ensure a sufficiently large torque is applied. A rigid and robust aluminium frame was made. A simulation of the final system model showed a working system, which is able to balance stably and compensate an impulse disturbance.

5

Results

The self-balancing robot was designed and realized according to Chapter 4. The simulation showed a stable self-balancing robot and in this chapter the results of the realized robot are described.

5.1 Achievements

The end result of the project is shown in Figure 5.1; the realized self-balancing robot.
As shown, the system was designed and realized completely standalone, i.e. no external components are required to run the system. Also, the robot is built rather robustly and is quite easy to use, as there is only a power switch (which activates the system) and a reset button (which resets the microcontroller). Thus, the first three specifications, as stated in Section 2.1, are met.

Besides this realization, a bond graph model was constructed which is a good starting point for future work, e.g. improvements to the system.

5.2 System Performance

The system performance is described qualitatively in terms of stability and compensation to a disturbance. Also, an estimation of the time of operation is given.

5.2.1 Stability and disturbance compensation

On a flat floor, the system balances itself stably. A small vibration can be seen, but the angle of the bar always remains approximately zero. This vibration is possibly caused by a little play between the two spur gears.

A 'one-finger' impulse disturbance to the top of the bar is compensated to obtain a bar angle of approximately zero again. After this compensation, the forward velocity of the robot is zero. These results are in accordance with the simulation result of Section 4.5.

At least the fourth and fifth specification are met (see Section 2.1). It was not measured whether the last specification was met, but it was seen that it is approached.

5.2.2 Time of operation

The time of operation when powered by batteries was not measured precisely. However, the batteries lasted one entire day of testing without noticing any performance problems. The time each test took varied greatly. As an estimation, a continuous operation of the system of at least 30 minutes is given. This corresponds to about 15 demonstration runs, assuming each demonstration takes about 2 minutes.

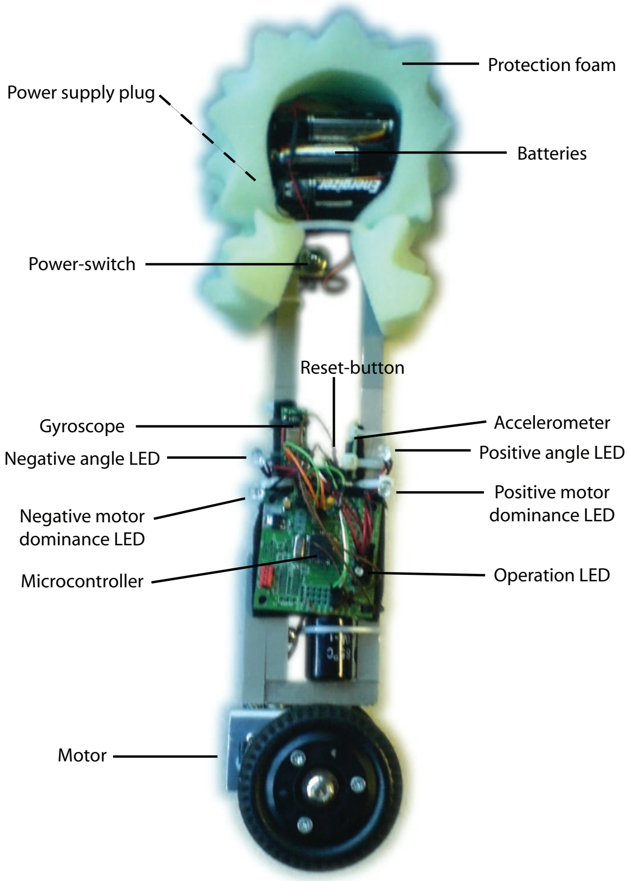
Protection foam
Power supply plug
Batteries
Power-switch
Reset-button
Gyroscope
Accelerometer
Negative angle LED
Positive angle LED
Negative motor dominance LED
Positive motor dominance LED
Microcontroller
Operation LED
Motor

Figure 5.1: End result of the project

5.3 Reflection

The power supply as well as programming and debugging the embedded system appeared to be difficult aspects in the realization of the robot.

It was difficult to realize a stable 5 V power supply for the embedded system and sensors, while a motor is continuously drawing 751 mA from the same batteries at the same time. When the batteries are fully charged, no problems occur. However, when the batteries are running low, a drop of the supply voltage, as a result of the motor suddenly drawing current when the rotation is reversed, can be measured at the output of the sensors, which influences the measurements and thus the balancing performance. A voltage drop of 43 mV, from 4.98 V to 4.937 V, was measured. The sensor outputs, i.e. the A/D-inputs, also dropped measurably by this much. Because the LSb of the A/D-converter represents 4.88 mV at the input, the digital output value is $\frac{43}{4.88} = 8.8$ lower than it should be. From Section 4.2.3 it follows that the accelerometer measurement is then short of $8.8 \cdot 0.0041 \approx 0.036$ rad. The gyroscope measurement is short of $8.8 \cdot 0.017 \approx 0.15$ rad/s. The calculated angle will be less than the actual angle and because the angle of the bar should be close to zero when balancing, the calculated angle will most likely be negative. The controller reacts in such a way that the motor will have a rotational dominance in the negative direction, which causes the upright standing bar to fall in the positive direction. This is then compensated, but a tendency to fall in the positive direction persists.

As no 20-Sim code generation library was present for the Atmel microcontroller, the software was written manually, which appeared to be troublesome. No knowledge of the Atmel microcontrollers was gathered in the past, so problems could not be solved quickly and easily and a lot of debugging had to be done to ensure correct behavior.

6

Conclusion

6.1 Conclusions

As was elaborated in this report, a working self-balancing robot was designed and realized during this project. Most of the specifications that were made have been met and the robot can be used as a demonstration model, which was the aim of the project. The performance results of the actual robot correspond to the simulation results. It is concluded that the model of the system gives a good representation of the actual behavior. The model and this system can act as a starting point for future work, e.g. improvements can be added (Section 6.3) or a new person-carrying self-balancing robot can be designed.

Section 5.3 described the difficulties while realizing the robot. The power-supply can be improved relatively easy, as is described in Section 6.2. However, manually writing software for the microcontroller (or the embedded hardware in general) should not be necessary during the Mechatronics-project. Debugging program code is very time consuming and is not very insightful. When the code is generated automatically, more attention can paid to other things, e.g. optimal controller design. Thus, as a recommendation for the Mechatronics-project, a 20-Sim code-generation library for the embedded hardware should be made available.

6.2 Design Reconsiderations

Although the system works well and as intended, it turned out that some aspects could have been done better.

- As was described, a stable power supply was difficult to realize. A motor suddenly drawing much current, e.g. when the rotational direction is reversed, resulted in a voltage drop, which caused an incorrect angle measurement and thus degraded system performance. When using one 9 V supply for the motor and a separate 5 V supply for the embedded system and the sensors, a stable 5 V supply can be realized.

- The use of two H-bridges placed in parallel was needed as a ‘work-around’, because one IC could not deliver enough current, due to an error in the datasheet. A stronger H-bridge can solve this ‘work-around’, for approximately the same price.

- The motor is driven by a PWM-signal. This means the motor is always rotating, even when no compensation is needed, as the duty-cycle then is 50 %. This causes small vibrations, that reduce the performance of the system. Furthermore, current is always drawn from the power supply, which is not very energy efficient. A higher PWM-frequency can help to reduce vibrations and shutting down the PWM-signal when the duty-cycle is 50 % will reduce power consumption. Another possibility is to drive the motor with a varying current instead of the on/off-like way when using a PWM-signal. Probably, a D/A-converter is then needed to convert a digital value to an analog current, as well as some electronics to take care of reversing the rotational direction of the motor.

6.3 System Improvements

Future improvements can be added to this system, to make it even more interesting.

A nice improvement would be to implement a way of driving and steering the robot, e.g. a line-following robot. Two motors are then needed to be able to steer, as well as sensors to respond to the environment, e.g. object detection. The existing bond graph model probably needs to be adjusted in such a way that an extra dimension is added (y-direction), as now only the x and z-direction are taken into account.

A next step could be to be able to navigate the robot, e.g. a radio-controlled robot which can be navigated by a user.

Appendix A

Introduction to Planar Mechanics

The planar mechanics description was used to create a bond graph model of the system. This appendix gives an introduction to modeling planar mechanics, taken from [1].

Planar mechanics are idealized systems that can translate and rotate in one plane (2D) instead of in space (3D). Only two translational directions and one rotation around one axis, perpendicular to this plane, can take place. These limitations are called constraints.

The essential problem of modeling multi-dimensional mechanical systems is the description of the so-called kinematic relations, determined by the geometry of the mechanism. The geometry is described in terms of displacement variables, i.e. displacement with respect to a previously defined reference point. All displacements are defined with respect to this point of reference, which is in fact a reference frame. This is often an inertial frame, or non-accelerating frame, in order for Newton's second law ($F = \frac{\mathrm{d}p}{\mathrm{d}t}$) to hold.

The idealized system consists of one or more rigid bodies; an idealization of a solid body of finite size in which deformation is neglected. Each rigid body has its own body-fixed frame that can be chosen freely, but the position of the origin of the frame is often placed in the center of mass. Rotation and translation of a point in this body can be described with this frame as a reference. The position of the origin and the orientation of its axes are described using the reference frame. In Figure A.1, a rigid body with a body-fixed frame is shown. The reference frame is shown as well.

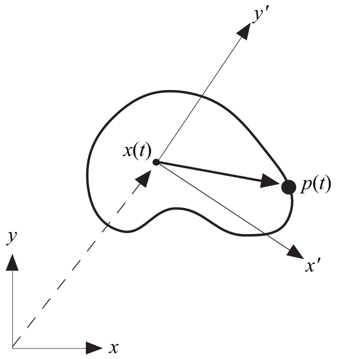

Figure A.1: Rigid body with body-fixed frame in a reference frame

To illustrate the description of planar mechanics, an example is given. Figure A.2 shows an inverted pendulum consisting of a mass m, subject to gravity g, at the top of a massless bar.

The other end of the bar is connected in a hinge point which allows in-plane rotation, in which some friction is assumed. In this simple example, translational motion of the hinge point is not discussed.

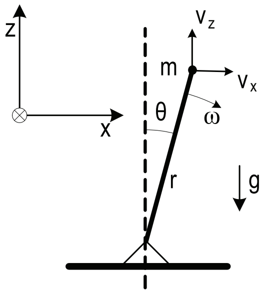

Figure A.2: Inverted pendulum consisting of a point mass connected to a massless bar in a hinge point

It can be seen that there are two domains (rotation and translation). The gravity can be represented by an Se-type source, the point mass by an I-type storage element. This storage of kinetic energy and the gravitational port can be described in a Cartesian coordinate frame (translation in x and z-direction), in which one axis is parallel to the gravitational acceleration and, thus, the other perpendicular. A difference between this case and linear translation is that there are now two I-type storage elements representing the point mass m: one for the x-axis and one for the z-axis. The position of the mass can be described using polar coordinates in the rotation domain, in which the position of a point is described in terms of the distance r to the origin and a rotational displacement θ. Inertia should be present in the rotation domain, but as a point mass has no inertia, it is omitted. Bearing friction of the hinge point only concerns the angular velocity, so this friction is added too. When taking air friction into account, it should be described in the translation domain with identical resistive ports in both directions.

Kinematic relations can be derived; a coordinate transformation from polar to Cartesian coordinates.

$$x(t) = r \cdot sin\,\theta \tag{A.1}$$

$$z(t) = r \cdot cos\,\theta \tag{A.2}$$

These equations can be described in flow variables by differentiating with respect to time ($\dot{\theta}(t) = \omega(t)$).

$$\dot{x}(t) = r \cdot cos(\theta) \cdot \omega \tag{A.3}$$

$$\dot{z}(t) = -r \cdot sin(\theta) \cdot \omega \tag{A.4}$$

Or in vector notation:

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} r \cdot cos\,\theta \\ -r \cdot sin\,\theta \end{bmatrix} \cdot \omega \tag{A.5}$$

This is a relation between two flow variables (translational and angular velocity), which can be represented by a power continuous multiport which is state modulated by the state variable θ, i.e. an MTF-element. This multiport transformer can also be decomposed into two-port MTF-elements, resulting in a single-bond notation. Figure A.3 shows the resulting single-bond bond graph of the inverted pendulum.

Figure A.4 shows a simulation of the inverted pendulum model. The initial angle was set to 0.0175 rad and it is shown that the pendulum falls and oscillates around π rad. The pendulum moved through the ground, as no ground layer was implemented, and hangs upside down in a stable position.

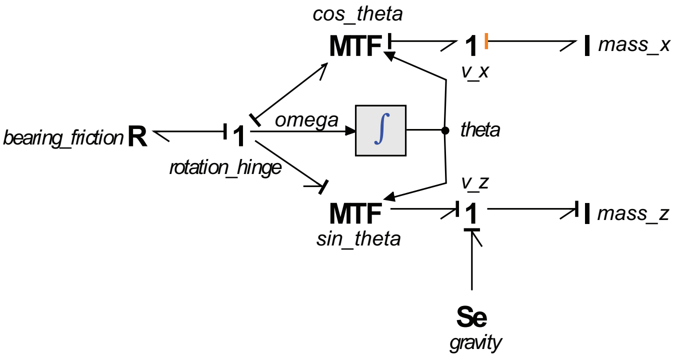
cos_theta
MTF
1
mass_x
v_x
bearing_friction R
1
omega
∫
theta
rotation_hinge
v_z
MTF
1
mass_z
sin_theta
Se
gravity

Figure A.3: Bond graph model of the inverted pendulum

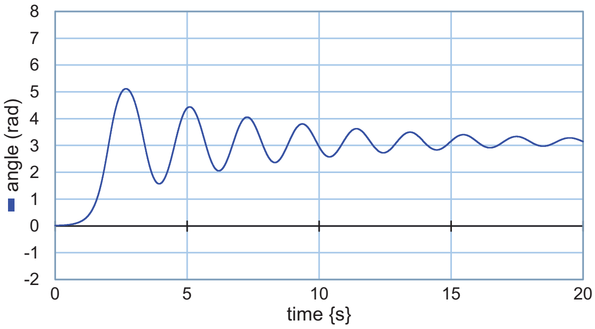

Figure A.4: Simulation of the inverted pendulum model

Appendix B

Diagrams and Schematics

Figure B.1 shows a flow diagram of the code which is run by the microcontroller.

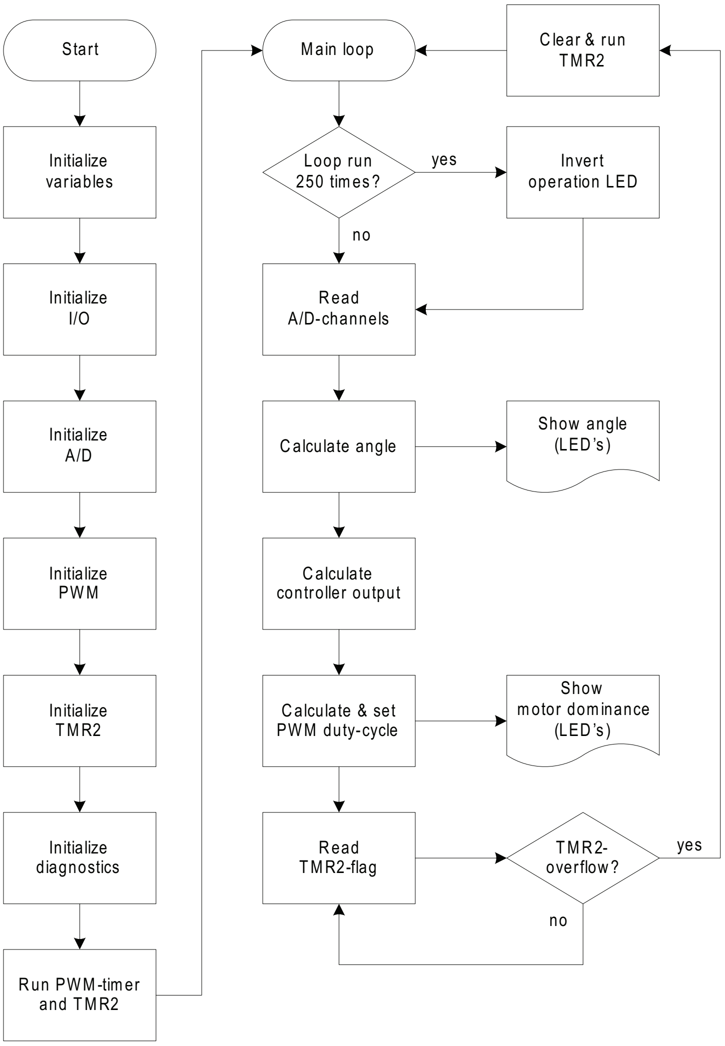
Start
Initialize variables
Initialize I/O
Initialize A/D
Initialize PWM
Initialize TMR2
Initialize diagnostics
Run PWM-timer and TMR2
Main loop
Clear & run TMR2
Loop run 250 times?
yes
Invert operation LED
no
Read A/D-channels
Calculate angle
Show angle (LED's)
Calculate controller output
Calculate & set PWM duty-cycle
Show motor dominance (LED's)
Read TMR2-flag
TMR2-overflow?
yes
no

Figure B.1: Flow diagram of the code run by the microcontroller

In Figure B.2 a schematic of the electronics can be seen. Where applicable, the component numbering is in accordance with the MiniMegaBoard. One port of the microcontroller has three holes on the MiniMegaBoard (microcontroller pin, ground and V_{bat} or V_{dd}). The hole closest to the inside is wired to the microcontroller.

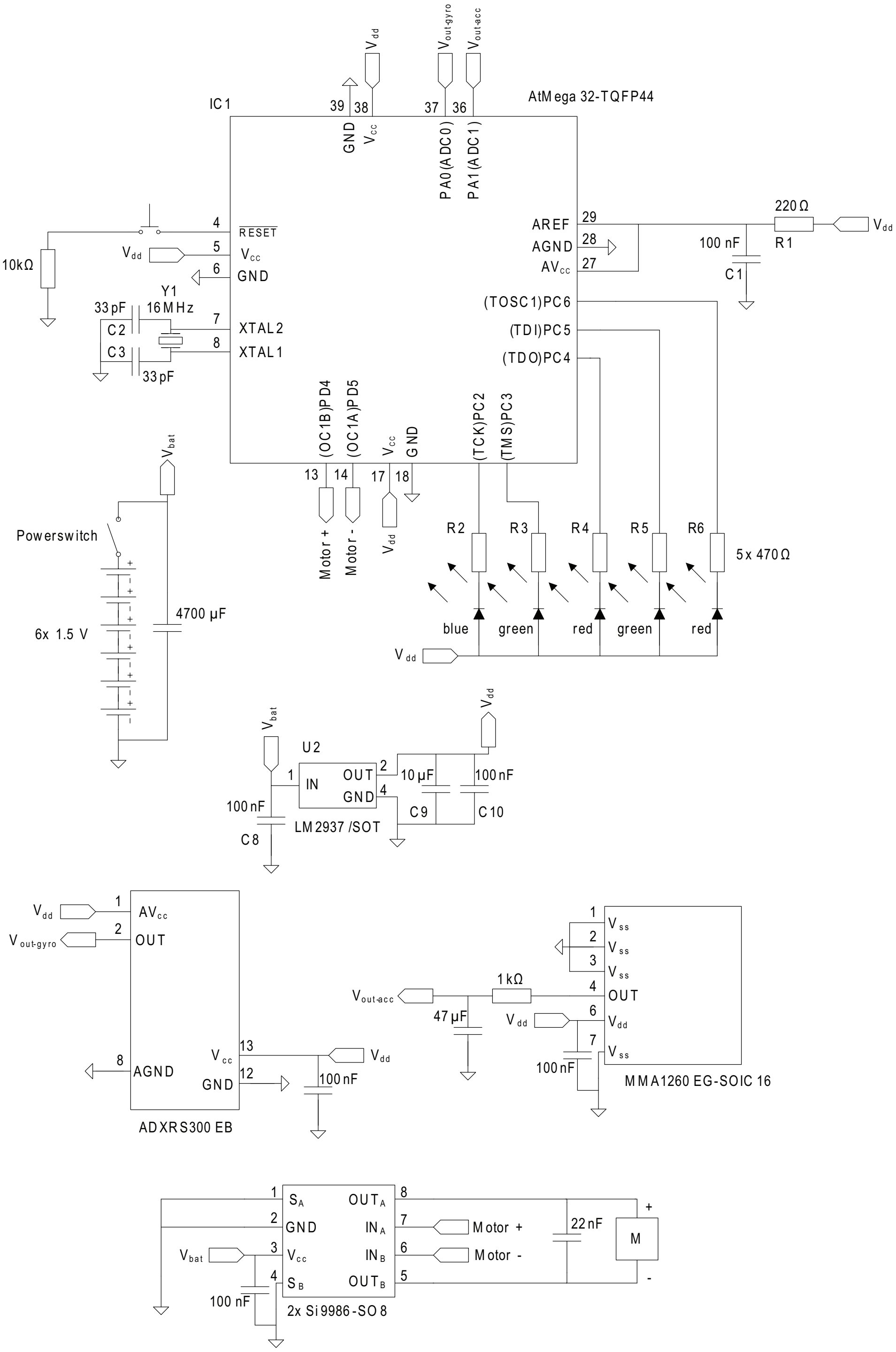
IC1
AtMega 32-TQFP44
Vdd
Vout-gyro
Vout-acc
39 38 37 36
GND Vcc PA0(ADC0) PA1(ADC1)
4 RESET
5 Vcc
6 GND
7 XTAL2
8 XTAL1
10kΩ
Vdd
Y1 16MHz
33pF C2
C3 33pF
29 AREF
28 AGND
27 AVcc
220Ω R1
100 nF C1
Vdd
(TOSC1)PC6
(TDI)PC5
(TDO)PC4
(OC1B)PD4 (OC1A)PD5 Vcc GND (TCK)PC2 (TMS)PC3
13 14 17 18
Motor + Motor - Vdd
R2 R3 R4 R5 R6
5x 470Ω
blue green red green red
Vdd
Vbat
Powerswitch
6x 1.5 V
4700 μF
Vbat
U2
1 IN
2 OUT
4 GND
LM2937 /SOT
100nF C8
10μF C9
100nF C10
Vdd
Vdd 1 AVcc
Vout-gyro 2 OUT
8 AGND
13 Vcc
12 GND
Vdd
100nF
ADXRS300 EB
1 Vss
2 Vss
3 Vss
4 OUT
6 Vdd
7 Vss
Vout-acc
1kΩ
47μF
Vdd
100nF
MMA1260 EG-SOIC 16
1 SA
2 GND
3 Vcc
4 SB
8 OUTA
7 INA
6 INB
5 OUTB
Vbat
100 nF
2x Si9986-SO8
Motor +
Motor -
22nF
M
+
-

Figure B.2: Schematic of the electronics

Figure B.3 shows a constructional drawing of the mechanics of the system. The frame is made of aluminium and the wheels are made of plastic with rubber tires around them.

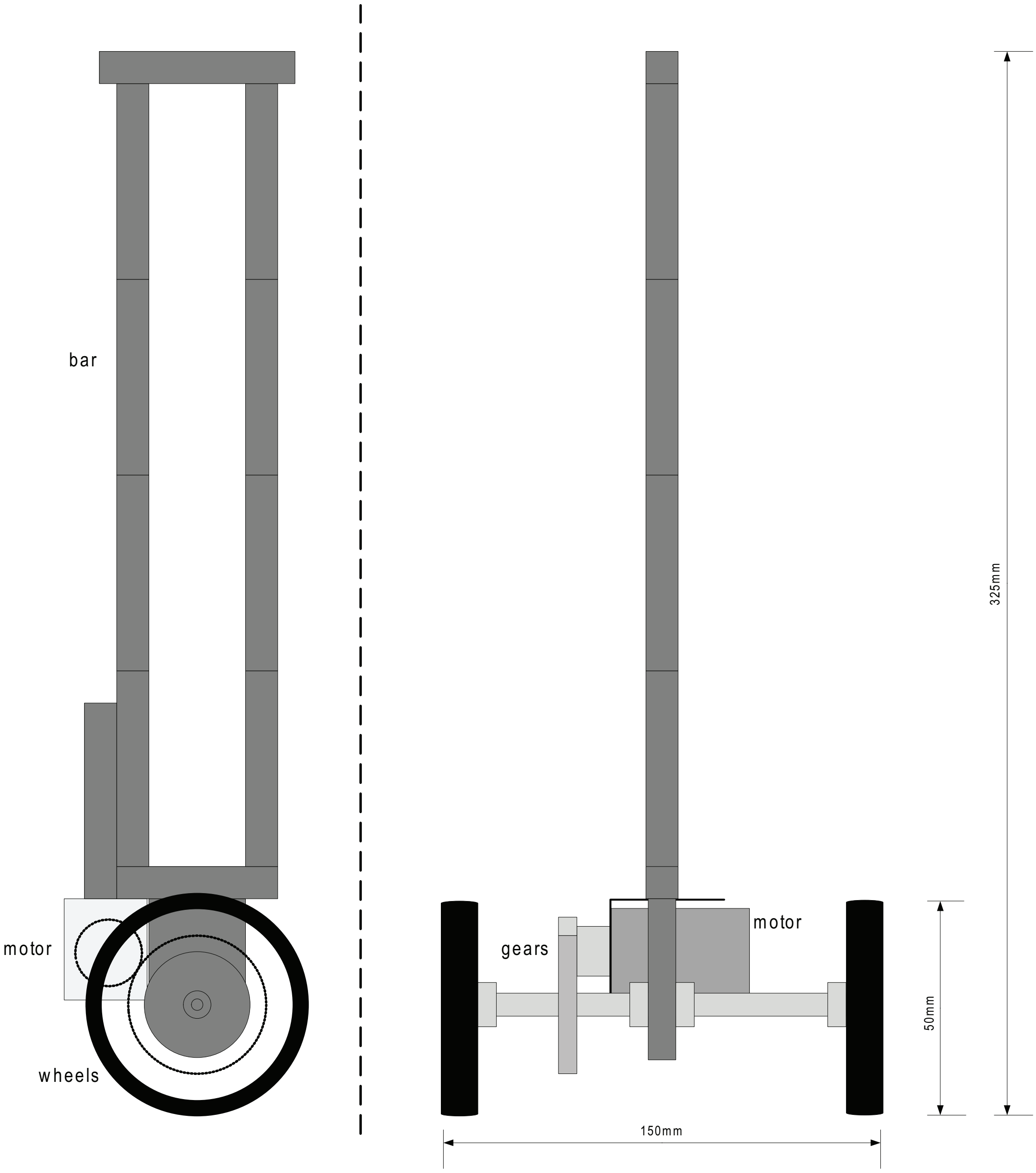
bar
motor
wheels
gears
motor
325mm
50mm
150mm

Figure B.3: Side view (left) and front view (right) of the system

Appendix C

Quick Reference Manual

This quick reference manual explains the key features in operating the robot, which is described step by step.

C.1 Features

The robot is operated by one power-switch and one reset-button. The power-switch is located at the top of the robot, right beneath the batteries. When the lever is pulled upwards, the power supply is connected to the system. The black reset-button is soldered to two brown wires, soldered to the PCB. While the reset-button is pressed, the microcontroller is in the reset-state and thus not executing any code. When the button is released, the microcontroller starts executing code again from the beginning of the program.

Two power supplies can be used, i.e. six 1.5 V batteries and an external 9 V supply which can be plugged in. Both are connected to the system, when using the power-switch.

Take care to remove at least one battery when using the 9 V supply!

C.2 Operation

The following steps should be taken to operate the robot:

1. Place the robot on a flat surface and keep holding it in the upright position as well as possible;
2. Switch on the power by using the power-switch, the system is now starting up;
3. All LED's will flash five times to show the microcontroller is ready and starting to execute the main loop;
4. After flashing its LED's, the system will balance itself, so there is no need to hold it anymore.

The robot is protected by some blue 'foam', so it will not be damaged when it might fall over. When the robot has fallen, the following should be done:

1. All LED's will flash to show the microcontroller has detected a fall and stopped executing any code;
2. The reset-button should be pressed and hold to enter the reset-state;
3. Again, place the robot on a flat surface in the upright position;

4. Release the reset-button and wait for the flashing LED's, after which the robot can be released.

As explained in Section 5.3, the system performance degrades when the batteries are running low. Either new batteries should be used or the 9 V supply should be plugged in, when experiencing performance problems.

Figure 5.1 shows the system with the placement of the most important components.

References

[1] Breedveld, P.C., *Integrated modeling of physical systems*, Planar mechanisms: Common equation formulation and bond graphs, Preliminary version, 1998

[2] Breedveld, P.C. and J.F. Broenink, *Mechatronica ontwerptraject 2007/2008*, 2007

[3] Control Lab Products, *20-Sim*, 2002

[4] Dertien, E., *MiniMegaBoard*,
https://cewiki.ewi.utwente.nl/wiki/index.php/MiniMegaBoard

[5] van Kuppeveld, T., *Model-based redesign of a self-balancing scooter*, MSc Report, August 2007