TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

Practical

Line Following Robot

Documentation

Lukas Silberbauer
e0126310@student.tuwien.ac.at

7th January 2005

Contents

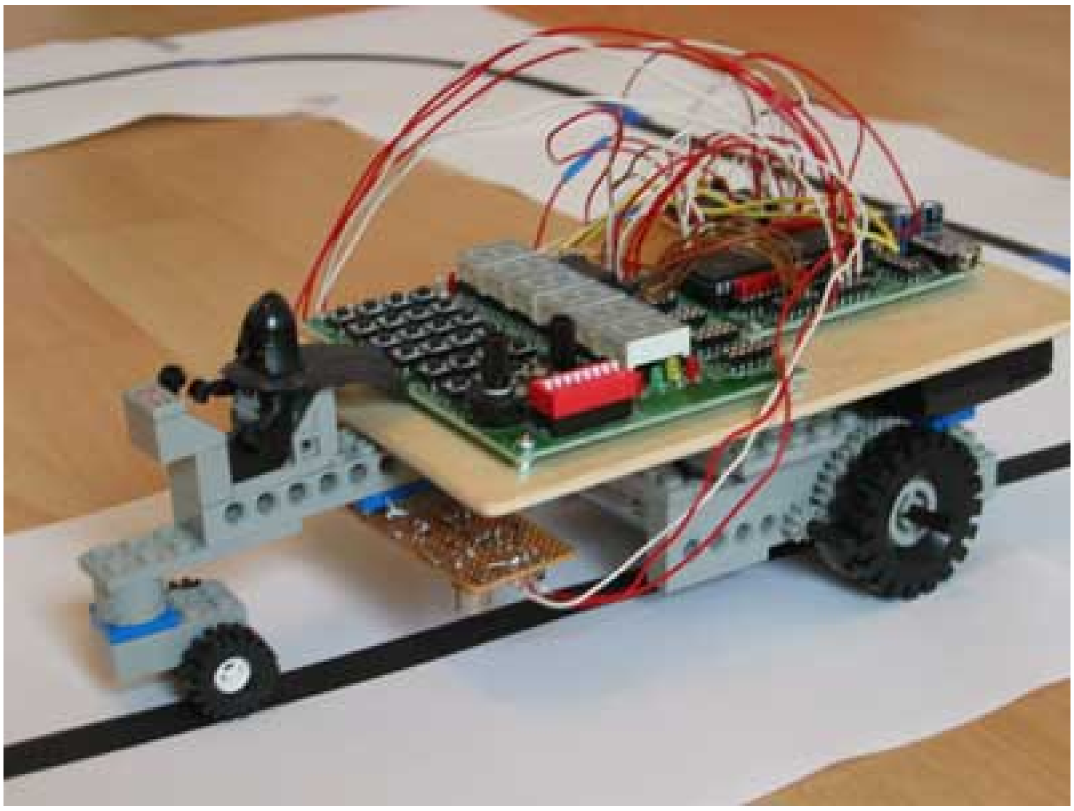


Figure 1: first working prototype - July 7th, 2004

1 Introduction

1.1 Preface

This practical has been established to provide the microcontroller course at the Vienna University of Technology with an autonomous robot. The robot should be programmed by students participating in this course. The goal of this practical is to develop a working prototype suitable for teaching purposes.

Line following is the ability of an autonomous robot to follow a line marked along the floor. This primary objective should be accomplished in the least amount of time.

1.2 Requirements

These features are mandatory for the robot:

- high extensibility
- low complexity
- low costs

1.3 Goals / Aims

The following points should be heeded to guarantee the best possible acceptance:

- easy to program
- high speed and maneuverability
- cool exterior

1.4 Existing Systems

Line following is a popular topic many robot engineers already dealt with. Therefore several competitions are held worldwide among line following enthusiasts each year. Many successful projects are well documented available through the www.

An extensive research effort has been undertaken to evaluate different solutions and to avoid design mistakes. Here is a list of ideas gathered during the web research phase:

From `http://www.robotroom.com/Sweet.html`:

- Lego compatible shaft for different types of wheels
- eventually place a bargraph on the front (debugging, fun)
- visible light sensors better than IR (tape lines cause trouble)

From `http://www.robotroom.com/Sandwich.html`:

- fancy headlights, nice exterior (chassis)
- white leds as light source improve different color tracking

From `http://elm-chan.org/works/ltc/report.html`:

- smooth steering algorithm

From `http://www.barello.net/Papers/LineFollowing/`:

- algorithms

From `http://www.seattlerobotics.org/encoder/200106/linerigel.html`:

- sensor tips, sample time, noise reduction, data processing algorithms, fast robot!

From `http://www.wa4dsy.net/robot/line/`:

- schematics and source codes

From `http://www.kmitl.ac.th/~kswichit/LFrobot/LFrobot.htm`:

- award winning robot, very slow though

1.5 Application Boundaries

There are many tradeoffs one has to face while designing a robot. For example there is conflict between speed and durability, because heavy batteries improve range while reducing the robots maneuverability. High speed turns are always limited by the grip of tires, because, most important, the laws of physics can never be broken.

2 Building the Prototype

2.1 Robot features

- 6 line tracking sensors
- 2 H-bridge motor controllers for 2 DC motors
- low dropout voltage regulator
- 8 debug leds (bargraph)
- play/pause switch
- RF transceiver

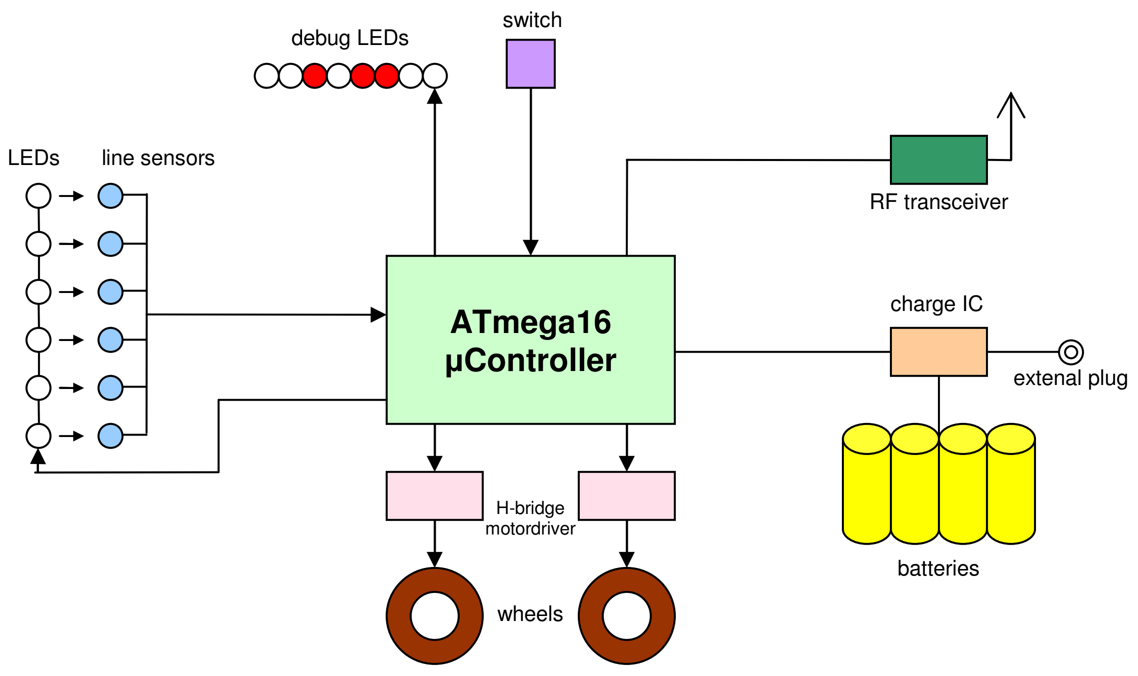


Figure 2: concept

2.2 Microcontroller

It seems practical to use the ATmega16 controller which is currently used in the lab. This controller is mounted on a multi-purpose Controller Board.

ATmega16 Features:

- Advanced RISC Architecture, 16 MIPS Throughput at 16 MHz
- 16K Flash, 512 Bytes EEPROM and 1K SRAM
- two 8-bit Timers, one 16-bit Timer
- 4 PWM Channels
- 8-channel, 10-bit ADC

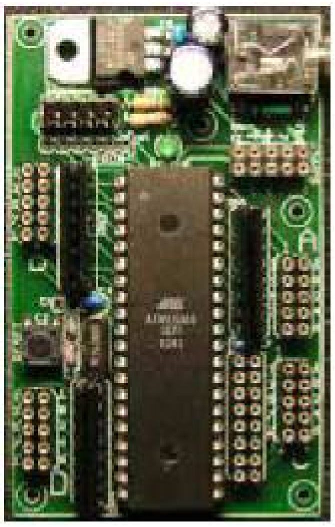


Figure 3: controller board

Totally it has 32 IO pins, which are protected by serial resistors on the Controller Board against short circuits.

2.3 Sensors

David Cook states on his homepage that photoresistors are not the best choice for line following due to their long reaction time. Phototransistors are fast and a 100 to 700 times more sensitive than photoresistors. Therefore an array of six BPY 62/III phototransistors is used. They "see" light between 420nm and 1130nm and have an half angle of 8 degrees.

Furthermore David Cook writes that visible-light emitters and detectors are superior to infrared for line detection. What looks like a line to a human eye may be almost transparent (masking tape) or completely opaque (certain clear plastics) to infrared. Since humans are building the courses, the robot should see using the same spectrum! Thus six SLH 36 WS white leds illuminate the path of our robot (visible light has a wavelength of 370nm - 750nm). They have an half angle of 25 degrees and a brightness of 300mcd.

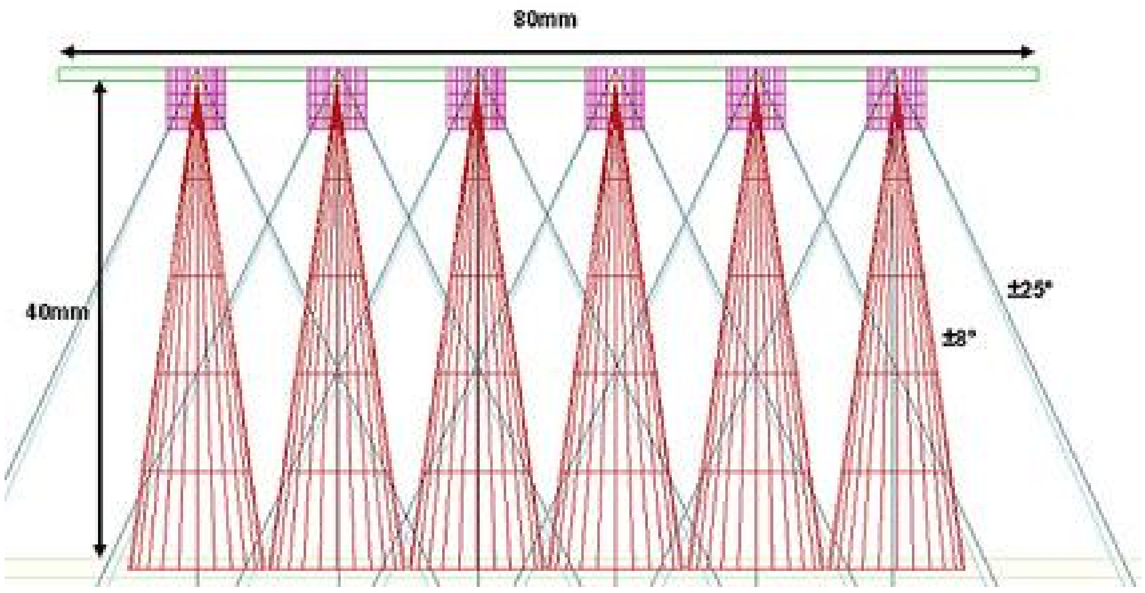


Figure 4: sensor sideview

Figure 4 shows the front view of our sensor-led arrangement. The sensors and leds are mounted on a 100x27mm printed circuit board. This board is 20 mm longer in reality than on this picture to hold the connectors.

Figure 5 demonstrates the alignment of the sensors. The surface with the 5mm line is located 40mm away from the PCB. Figure 6 shows that the real exposure corresponds with the model.

When creating the schematics for the sensor circuit this description of phototransistor circuits becomes handy. An interesting property of phototransistors is that their sensitivity is determined by the circuit in which the phototransistor is involved. With a high circuit resistance, the phototransistor has an increased sensitivity to light than with a low circuit resistance.

The actual resistor values are difficult to calculate, because the impact of ambient light is not known a priori. Therefore several experiments (see Figure 7, 8)are conducted to find the appropriate values. To sense the current flowing through the phototransistor a 100kOhm resistor is used. The difference between "line" and "no line" equals now 190bit with our ADC. Higher resistor values increase this difference but also increase the noise level on the lines.

In order to save energy the six leds don't shine all the time, they are switched by a 2N2218 transistor which draws only 4mA from the Microcontroller. With the constraints of our components in mind it is now possible to determine the maximum sampling rate of our robot, which could limit its speed. The rise/fall time of the BPY 62/III phototransistor is 7 us and our ADC has a Conversion Time of 65 - 260

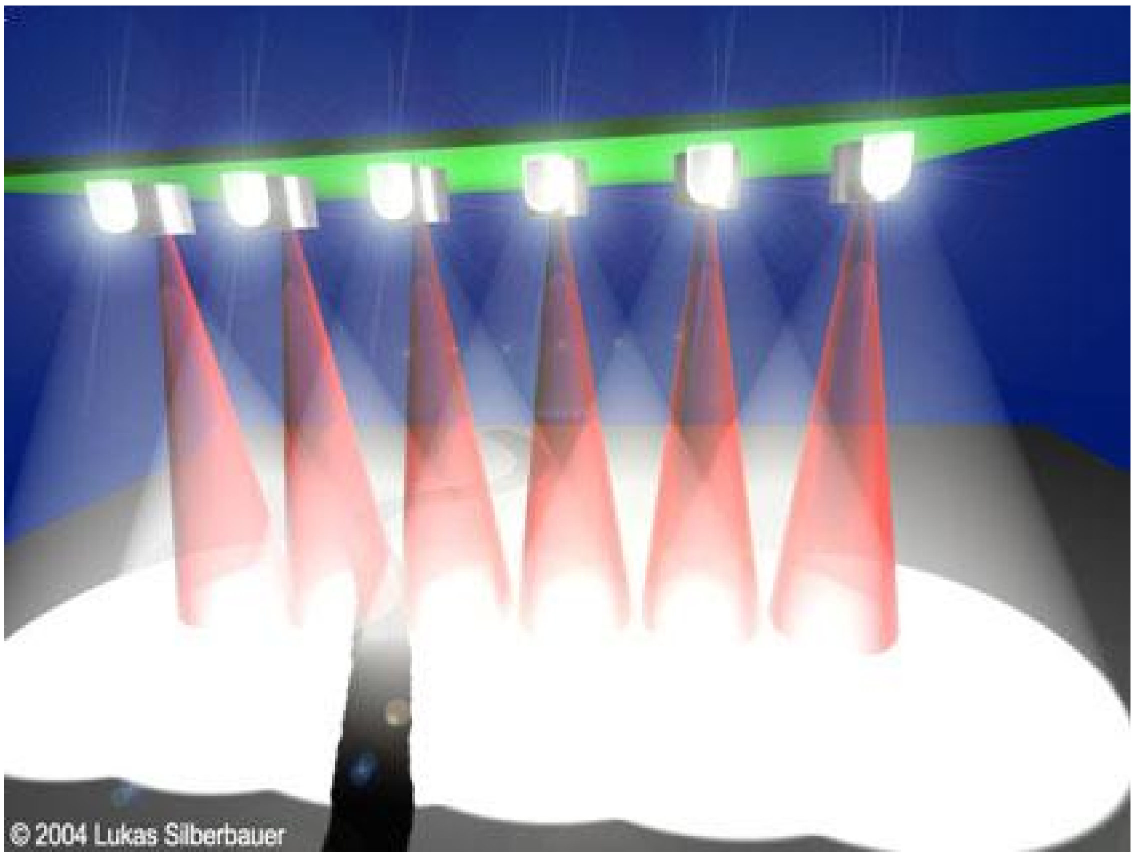


Figure 5: 3D Illumination Simulation

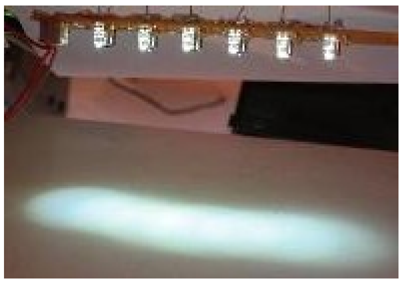


Figure 6: Illumniation Test

us, thus limiting the Sampling Rate of one sensor to 3.8 kHz (worst case, still not bad). Sampling all 6 sensors and finding the position of the line can therefore (theoretically) be done in 1.56ms at a rate of 641 Hz. Referring to a (hypothetically) 20 km/h moving robot that would be one sample every 8.6 mm. This example indicates that the real speed limiting factors are motors and traction rather that the sampling rate.

The next step was to test this sensor concept. Therefore the test circuit shown in figure 9 was used in connection with a HCS12 microcontroller for data acquisition.

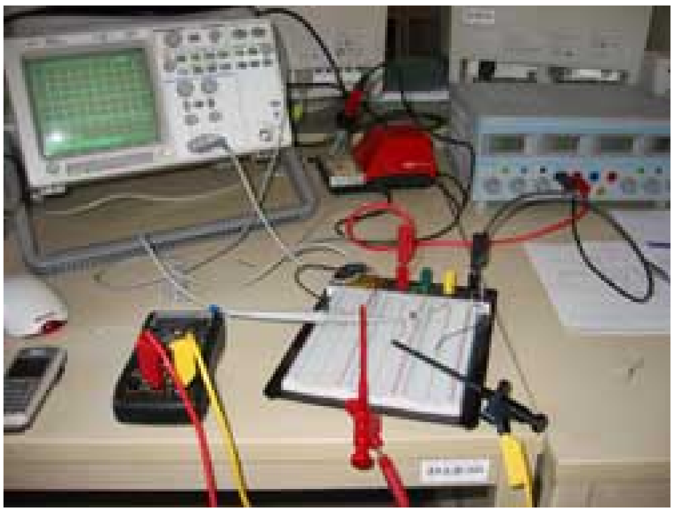


Figure 7: experimental setup

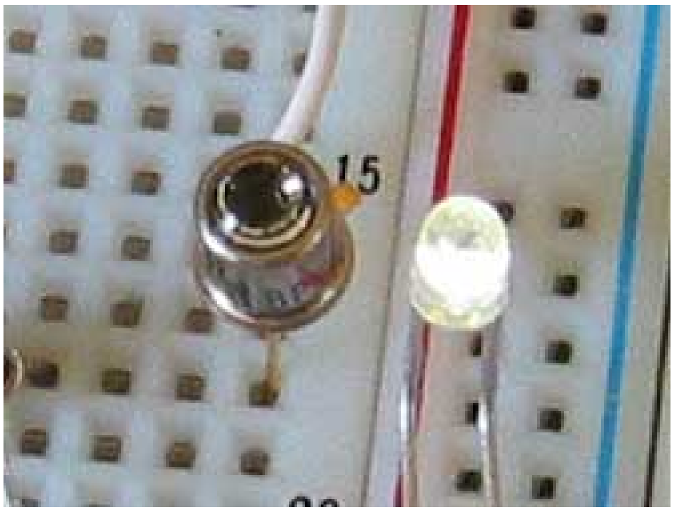


Figure 8: experimental setup

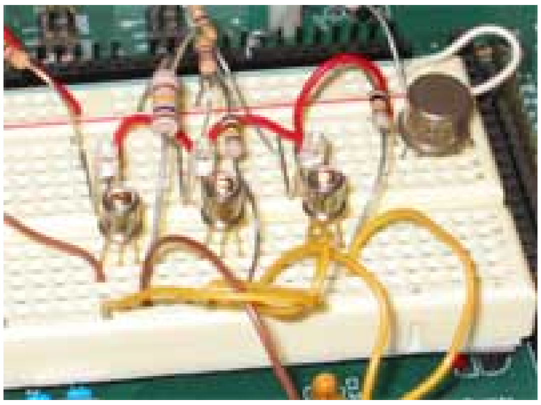
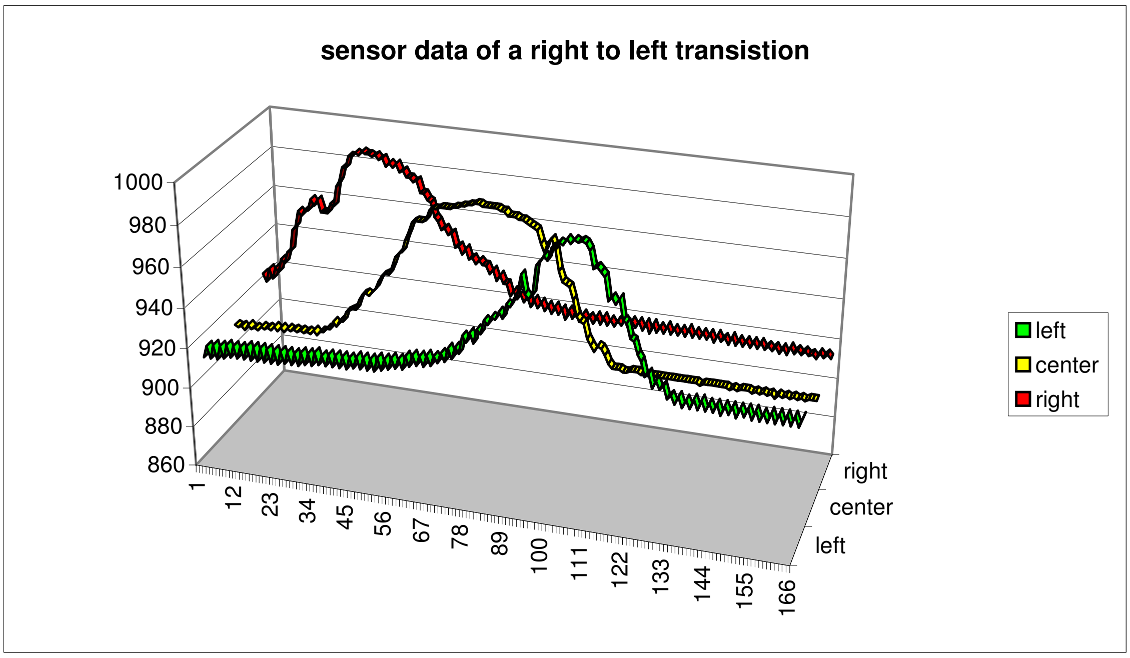


Figure 9: test circuit

The measured data was transferred to the pc via the RS232 interface and visualized using Excel. As figure 10 indicates, first tests look really promising. A white paper with a 12mm black line was pulled from right to left approximately 40mm over the sensor.



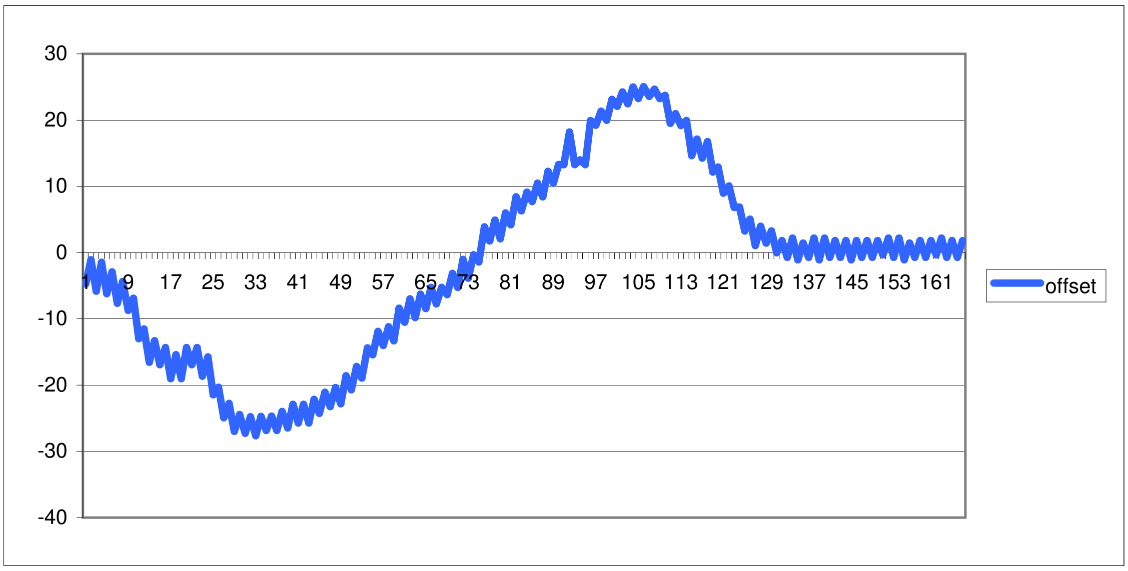


Figure 10: sensor data acquisition

Figure 11 shows the finished Line Sensor circuit soldered on a prototyping PCB. On the right there is a connector leading to the ADC inputs of the microcontroller. The silver cylinder on the bottom left is the 2N2218 transistor.

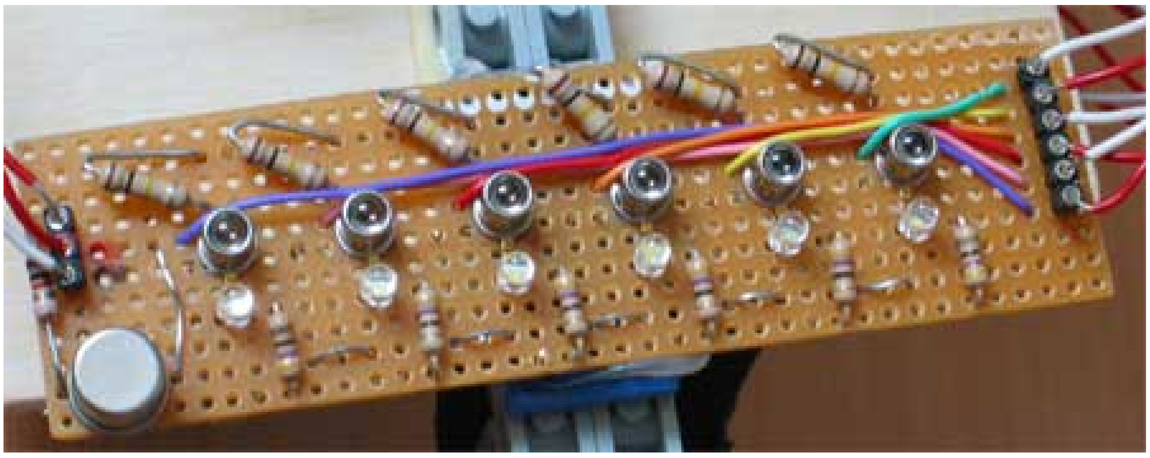


Figure 11: the sensor PCB

2.4 Motors

Our motors should produce much torque at low rpm and consume as little energy as possible. The following calculations were made to give us a raw idea of how much torque we would need to accelerate our robot (thanks to Markus Foltin).

$F = \frac{m*v}{t}$
(F... force needed to accelerate, m... mass, v... velocity, t... time)

$F = \frac{0.5*5}{3}$
(assuming 500grams and an acceleration from 0 to 5 m/sec in 3 sec)

$T = F * r$
(T... Torque, r... radius)

$T = 0.8333 * 0.02 = 0.016Nm = 16mNm$
(our wheels should have a diameter of 4 cm, friction is neglected)

4cm wheels would turn at 2387 rpm when driving 5m/sec. With this approximated data in mind it is possible to select appropriate motors for this special application. The IGARASHI 2430-65 motor (2.95 euro @ conrad.at) seems suitable. It produces 2.0 mNm at 7200 rpm, running at optimal efficiency. With a 1:3 transmission it should handle its job quite well (2 motors $\tilde{1}2$ mNm).

The motordriver L293D (equipped with internal clamping diodes) will be used, because our motor draws only a maximum of 1A current. First experiments with Lego showed that the robot would be way to fast with a 1:3 transmission, therefore a 1:25 transmission is used. However, our other assumptions were not too bad: the real robot weights 580 grams and the IGARASHI 2430-65 motors perform their job very well in accelerating the robot.

As power supply serves a NiCd 1100 mAh / 6 Volt battery pack from an old camcorder. It was chosen because it can be fast charged very easily with the existing battery charger of the camcorder. These 6 Volts go directly into the L293D dual H-bridge motordriver IC. A LP2950 low dropout voltage controller is used to feed the microcontroller and the sensor with 5 volt. The robot draws about 850 mA while driving, thus an operating time of 1.5 hours can be achieved theoretically.

Although both motors draw together only 650 mA and the L293D should handle up to 1.2A peak per channel it gets extremely hot. As soon as the L293D becomes too hot the robot slows down or stops completely, so cooling becomes an issue.

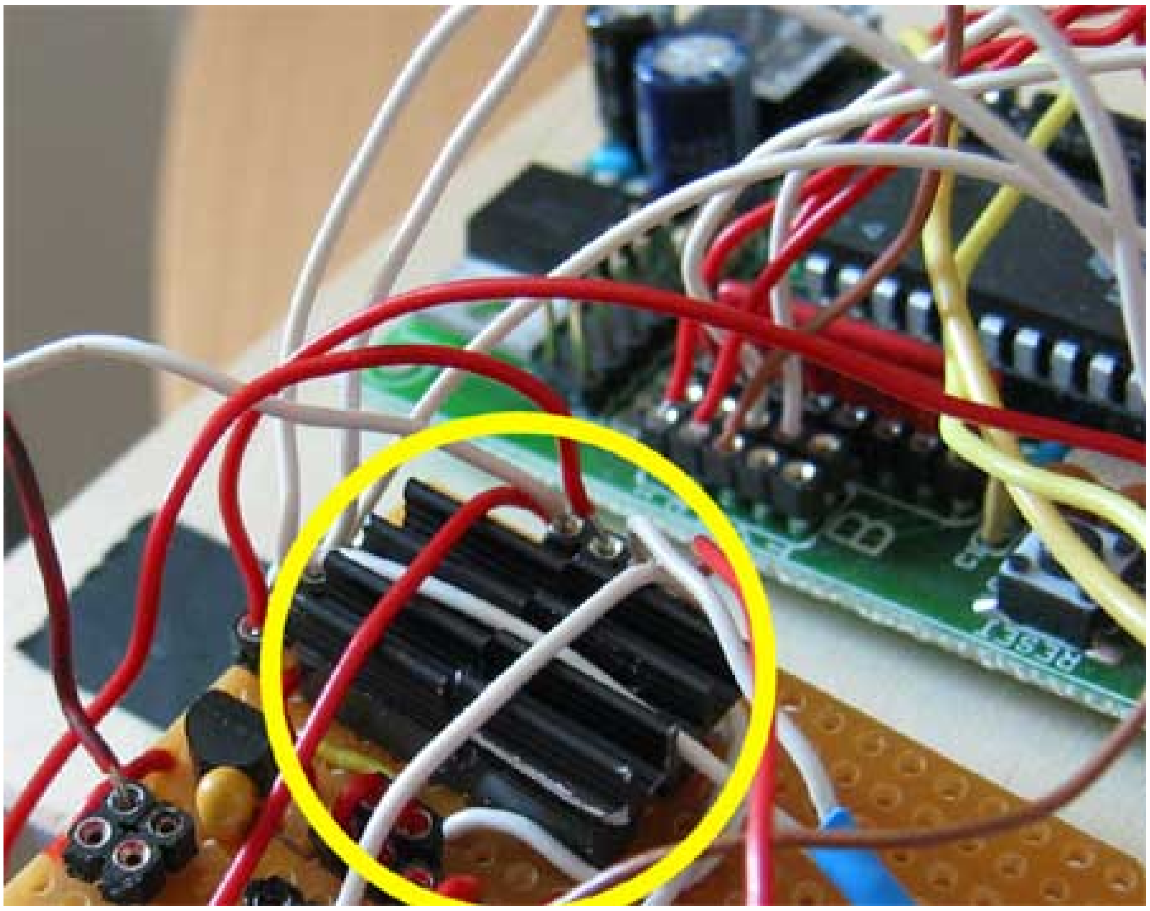


Figure 12: improvised cooler

Figure 12 shows a workaround for this problem. A custom made cooler was built out of the cooling fins of an old CPU cooler and attached to the L293D with wires. This solution renders it possible to drive around with 60 percent of the total speed without getting slowed down due to overheating. For faster speeds a ventilator would be needed.

2.5 RF transceiver

Equipping this robot with an RF transceiver and making it remote controllable is part of another practical. It should be possible to program the robot over the internet and watch its movements via a webcam. Once this is done, a link will be placed here.

2.6 Mechanical Assembly

Lego was used to build the robot because it is widely available and it is very easy to build transmissions. It is also quite stable if glue is used to hold the parts together. The microcontroller itself is mounted together with an IO board on a wooden panel, which was originally created for the microcontroller course. It is used unmodified for the robot, held in place by double sided adhesive tape. Although the IO board is not necessary for operation it is good to have it for debugging. Below the micro-controller is the H-bridge motordriver circuit (the brown PCB).

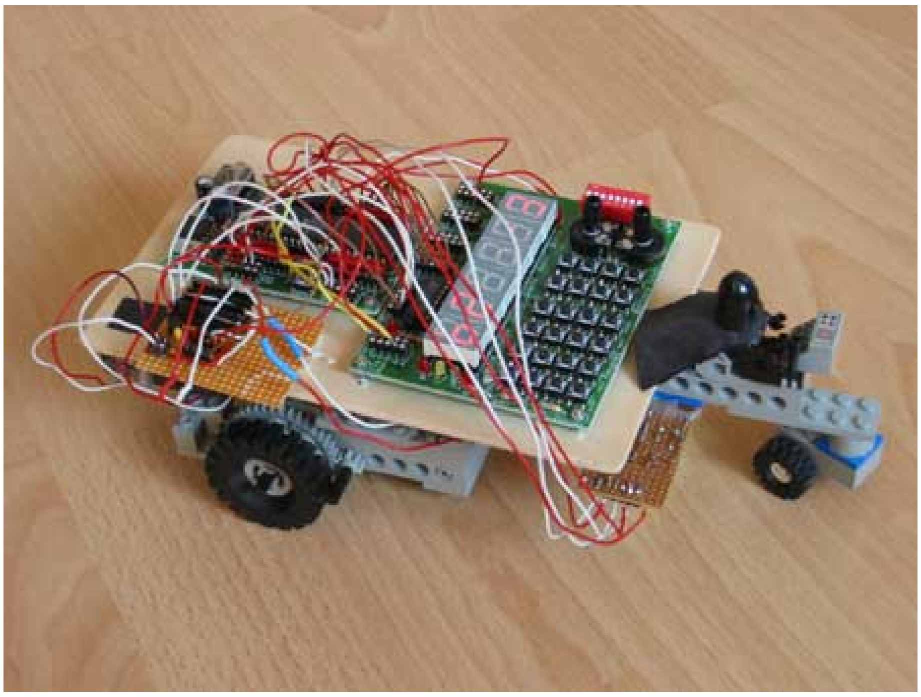


Figure 13: finished prototype

Two Lego parts where modified with the Dremel tool to hold the motors in position, which are attached with 2 component epoxy glue. Figure 15 shows the motors and the transmission. The gears directly after the motors are from a gear set, obtainable at conrad.at. The Line Sensor is attached to the Lego frame with hot-melt adhesive.

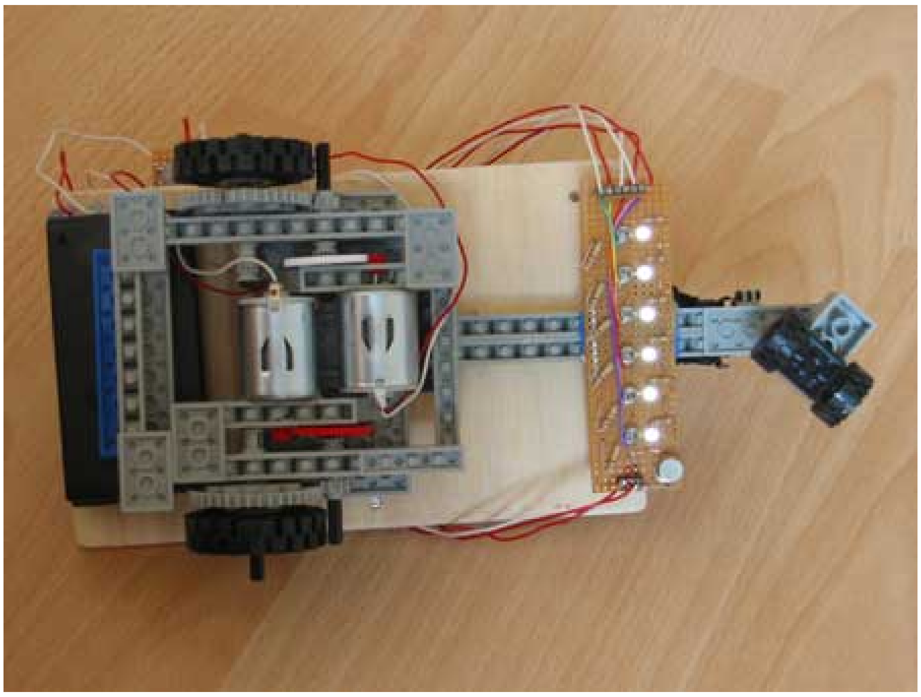


Figure 14: finished prototype

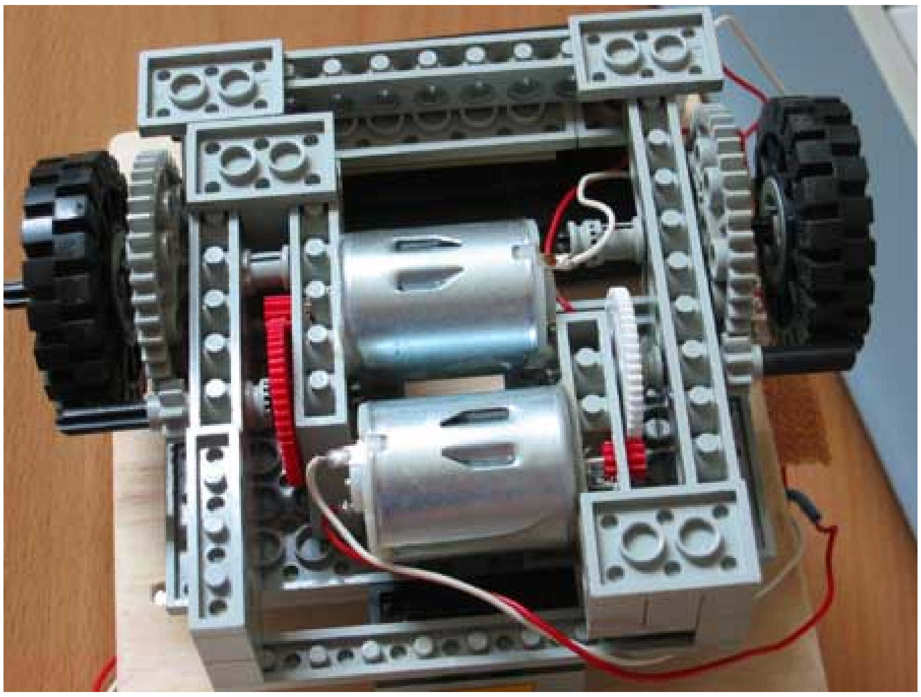


Figure 15: mechanical assembly

2.7 Application Software, Algorithms

As development environment WinAVR is used.

The first steering algorithm used is very simple, but effective. The maximum of the 6 sensor values is determined and interpreted as the position of the line. The two motors are driven by a PWM signal with a period time of 1msec. A simple switch statement does the steering:

```
/*
 * max_pos: position of the line (maximum of sensor values, 0 to 5)
 * drive_speed_m1: left motor speed (-100 to 100)
 * drive_speed_m1: left motor speed (-100 to 100)
 */


switch (max_pos)
{
case 0:
drive_speed_m1 =  60;
drive_speed_m2 = -20;
break;
case 1:
    drive_speed_m1 =  40;
drive_speed_m2 =   -5;
break;
case 2:
     drive_speed_m1 =  30;
     drive_speed_m2 =  20;
break;
case 3:
     drive_speed_m1 =  20;
drive_speed_m2 =  30;
break;
case 4:
     drive_speed_m1 =  -5;
drive_speed_m2 =  40;
break;
case 5:
     drive_speed_m1 = -20;
drive_speed_m2 =  60;
break;
}
```

Note: In this example only a small percentage of the maximum engine power is used.

3 PCB Design

3.1 Introduction

After the concept has been proved by the prototype, an application specific printed circuit board (PCB) was created for small series production. It should comprise all the useful features of the prototype while expunging its deficiencies (e.g. the weak motor driver). Ultimately, a few extra features where added to challenge the students programming the robot.

3.2 Features

- Atmel ATmega128 microcontroller
- 2x L298 4Amp dual H-bridge motor driver
- 2x temperature sensor for motor driver
- Line Sensor (6 white leds, 6 phototransistors), removable
- JTAG and ICSP interface
- ER400TRS RF transceiver
- RS232 driver
- high side current monitor
- low drop voltage regulator
- 2 buttons
- 2 potentiometers
- 8 leds
- high extensibility
- supply voltage 7.5V - 16V

The goal of the design process is to create a versatile, multi-purpose PCB for various robotic applications. Virtually all IO pins of the ATMega128 are accessible through female connectors. Theoretically it is possible to create an extension board and stack it on top of the robot board. Many parameters of the board's state are accessible by software, e.g. current consumption, battery voltage and the motor driver's temperature. On board IOs include 2 potentiometers for parameter tuning, 2 buttons and 8 leds, therefore facilitating debugging. As communication interface serves a RS232 plug to connect the board directly to a PC, as well as the ER400TRS RF module for wireless access.

The line sensing section is apart of the other sections, rendering it possible to detach it. E.g. it is possible to mount the line senor elsewhere than the robot board and connect it by a ribbon cable.

3.3 Schematics

The light edition of the Eagle Layout editor by CadSoft is used for drawing the schematics and creating the board's layout. This freeware program is considered as one of the best schematic editors, providing also a great deal of free part libraries. To improve readability most signals are named after their function.

See Figure 16.

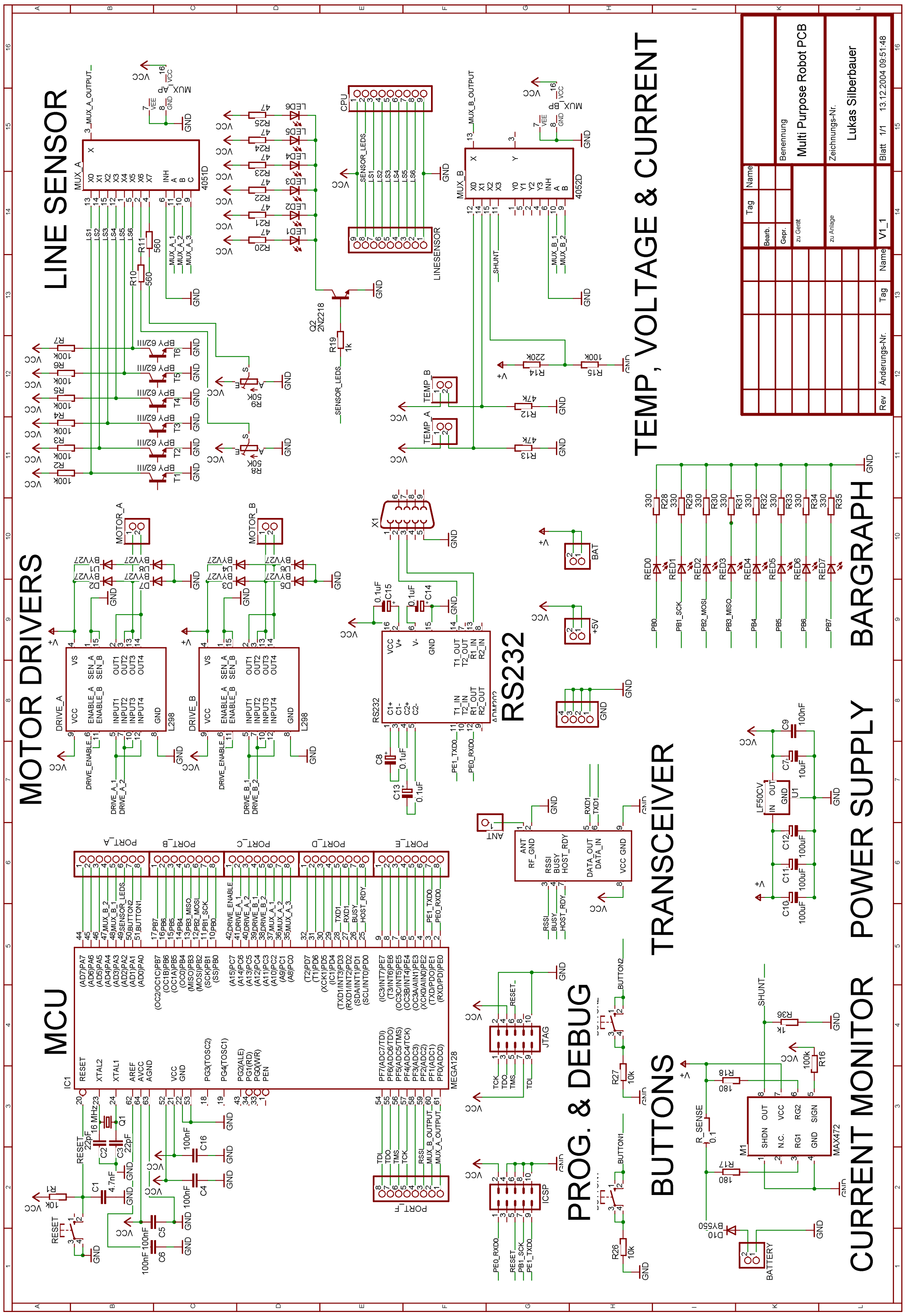


Figure 16: schematics

3.4 PCB Layout

For the board a 2 layer, half-euro sized printed circuit board is used. The biggest problem while generating the layout by placing the parts and drawing the wires is the lack of space. Therefore it is not possible to use one dedicated ground plane, some signals have to be routed on the ground plane layer. The board is produced by Olimex and fitted with components by hand.

See Figure 17.

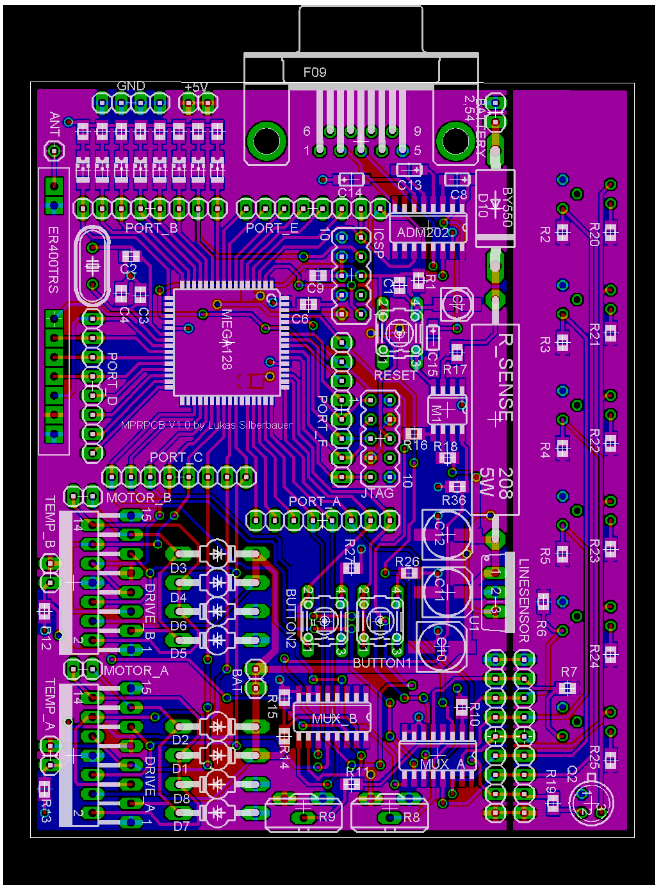


Figure 17: PCB Layout

3.5 Partslist

Most of the parts can be obtained directly from the German distributer Reichelt. The MAX472 was ordered as free sample at `www.maxim-ic.com`.

See Figure 18.

Quantity	Name	Package	Supplier	Order Number	Price (Qty 1)	Symbol(s)	Total
ICs							
1	ATMega128	TQFP64	Reichelt	ATMEGA 128-16 TQ	€ 10,50	IC1	10,50 €
2	L298	Multiwatt15	Reichelt	L 298	€ 2,70	DRIVE_A, DRIVE_B	5,40 €
1	74HCT4051D	SO16	Reichelt	74HCT 4051	€ 0,46	MUX_A	0,46 €
1	74HCT4052D	SO16	Reichelt	74HCT 4052	€ 0,28	MUX_B	0,28 €
1	ADM202	SO16	Reichelt	ADM 202 JRN	€ 1,95	RS232	1,95 €
1	LF50CV	TO-220	Reichelt	LF 50 CV	€ 0,77	U1	0,77 €
1	MAX 472	SO8	Maxim	Order No 445661 (2 fre	€ -	M1	- €
							- €
							- €
Div							
85	PINHEAD		Reichelt	Note: same as on MCLU - IO Board			- €
3	BUTTON	B3F-10XX	Reichelt	Note: same as on MCLU - IO Board		RESET, BUTTON1, BUTTON2	- €
2	ICSP, JTAG	MA05-2	Reichelt	Note: same as on MCLU - CPU Board		ICSP, JTAG	- €
1	CRYSTAL 16MHz	HC49/S	Reichelt	16-HC49U-S	€ 0,44	Q1	0,44 €
1	D-SUB 9 pol	F09HP	Reichelt	D-SUB ST 09EU	€ 0,27	X1	0,27 €
							- €
Capacitors							
1	4.7nF	C0805	Reichelt	X7R-G0805 4,7N	€ 0,05	C1	0,05 €
2	22pF	C0805	Reichelt	NPO-G0805 22P	€ 0,05	C2, C3	0,10 €
5	100nF	C0805	Reichelt	X7R-G0805 100N	€ 0,05	C4, C5, C6, C9, C16	0,25 €
1	10uF	153CLV-0405	Reichelt	SMD ELKO 10/16	€ 0,09	C7	0,09 €
4	0.1uF	CT3216	Reichelt	SMD TAN.0,1/35	€ 0,30	C8, C13, C14, C15	1,20 €
3	100uF	153CLV-0605	Reichelt	SMD ELKO 100/16	€ 0,17	C10, C11, C12	0,51 €
							- €
Diodes							
8	SMD-LED 1206 RT	CHIPLED_1206	Reichelt	SMD-LED 1206 RT	€ 0,11	RED0 : RED7	0,88 €
8	BYV27	SOD57-10	Reichelt	BYV 27/200	€ 0,20	D1 : D8	1,60 €
6	SLH 36 WS	LED3MM	Reichelt	SLH 36 WS	€ 0,54	LED1 : LED6	3,24 €
1	BY550	DO27-15	Reichelt	BY 550-50	€ 0,08	D10	0,08 €
							- €
Resistors							
3	10k	R0805	Reichelt	SMD-0805 10,0K	€ 0,10	R1, R26, R27	0,30 €
8	100k	R0805	Reichelt	SMD-0805 100K	€ 0,10	R2, R3, R4, R5, R6, R7, R15, R16	0,80 €
2	50K	PT-10S	Reichelt	PIHER 10-S 50K	€ 0,19	R8, R9	0,38 €
2	560	R0805	Reichelt	SMD-0805 560	€ 0,10	R10, R11	0,20 €
2	47k	R0805	Reichelt	SMD-0805 47,0K	€ 0,10	R12, R13	0,20 €
1	220k	R0805	Reichelt	SMD-0805 220K	€ 0,10	R14	0,10 €
2	180	R0805	Reichelt	SMD-0805 180	€ 0,10	R17, R18	0,20 €
2	1k	R0805	Reichelt	SMD-0805 1,00K	€ 0,10	R19, R36	0,20 €
6	47	R0805	Reichelt	SMD-0805 47,0	€ 0,10	R20, R21, R22, R23, R24, R25	0,60 €
8	330	R0805	Reichelt	SMD-0805 330	€ 0,10	R28, R29, R30, R31, R32, R33, R34, R35	0,80 €
1	SHUNT	KH208-8	Reichelt	5W AXIAL 0,1	€ 0,29	R_SENSE	0,29 €
							- €
Transistors							
6	BPY 62/III	TO18	Reichelt	BPY 62/III	€ 1,40	T1 : T6	8,40 €
1	2N2218A	TO18	Reichelt	2N 2218A	€ 0,30	Q2	0,30 €
						Total:	**€ 40,84**

Figure 18: partslist

4 The Art of Line Following

The algorithm used for the prototype is fairly simple, it is called discontinuous control algorithm. However, Line Following is a well-covered topic in literature, and there are some very interesting approaches in solving this, quite simple problem.

4.1 Discontinuous Controllers

A discontinuous controller is characterized by an output having two or more fixed states and its value is switched among these states according to the input value. The advantage of discontinuous controllers is that they are very easy to implement, the biggest drawback is that they don't suppress oscillating, which can be noticed at the Line Follower prototype.

A quite sophisticated approach can be found at `http://www.wrighthobbies.net/guides/linefollower.htm`. This robot, called Arty, uses 5 sensors. Although it does not use differential steering the code is very instructional. The reading of these 5 sensors is interpreted as a binary number, and different states can be distinguished:

00000 - Lost line from overshoot or break in line
00001 - Almost off the line, steer hard right and reduce speed
00011 - Near the right edge of the line, steer right
00010 - Right of the center of the line, steer right
00110 - Slightly to the right of the center of the line, slight correction to the right
00100 - Centered over line, increase speed for straight runs
01100 - Slightly to the left of the center of the line, slight correction to the left
01000 - Left of the center of the line, steer left
11000 - Near the left edge of the line, steer left
10000 - Almost off the line, steer hard left and reduce speed
11111 - Line intersection or circle at end of maze

The pseudo code looks as follows:

```
'       Lineflag is a variable that contains
'        a binary representation of the 5 sensors
'       Servo(1) is the steering servo PWM channel
'       Servo(2) is the drive motors PWM channel
'        controlled by an ESC
'       Overshoot is a flag that indicates if we
'        have lost the line at a turn
'      Bascom Basic Code Segment for Arty

Select Case Lineflag

Case &B00000 'No line
'If the bot loses the line, make steering changes
Servo(2) = Slow - Progspeed
If Lastlineflag < 4 Then
Overshoot = 1 'Overshot a righthand corner
Elseif Lastlineflag > 4 Then
Overshoot = 1 'Overshot a left hand corner
End If

Case &B00100 'Line under center sensor
Overshoot = 0
Servo(1) = Center
Servo(2) = Mediumfast - Progspeed

Case &B00110
Overshoot = 0
Servo(1) = Smallright
Servo(2) = Mediumfast - Progspeed

Case &B00010
Overshoot = 0 'Line under midright sensor
Servo(1) = Mediumright
Servo(2) = Mediumfast - Progspeed

Case &B00011 'Line between midright and right sensor
If Overshoot = 0 Then
Servo(1) = Largeright
Servo(2) = Medium - Progspeed
Else
Servo(1) = Mediumright
Servo(2) = Medium - Progspeed
End If

Case &B00001 'Line under right sensor
If Overshoot = 0 Then
Servo(1) = Hardright
Servo(2) = Mediumslow - Progspeed
Else
Servo(1) = Largeright
Servo(2) = Mediumslow - Progspeed
End If
```

```
Case &B01100 'Line between center and midleft sensor
Overshoot = 0
Servo(1) = Smallleft
Servo(2) = Mediumfast - Progspeed

Case &B01000 'Line under midleft sensor
Overshoot = 0
Servo(1) = Mediumleft
Servo(2) = Mediumfast - Progspeed

Case &B11000 'Line between midleft and left sensor
If Overshoot = 0 Then
Servo(1) = Largeleft
Servo(2) = Medium - Progspeed
Else
Servo(1) = Center
Servo(2) = Medium - Progspeed
End If

Case &B10000 'Line under left sensor
If Overshoot = 0 Then
Servo(1) = Hardleft
Servo(2) = Mediumslow - Progspeed
Else
Servo(1) = Largeleft
Servo(2) = Mediumslow - Progspeed
End If
End Select
```

4.2 PID based algorithms

Depending on how much effort one likes to spend, there are different ways to choose the right parameters for a PID based control algorithm. Mostly these values are found by trial and error, or following the Ziegler-Nichols method described here.

Because the modeling of a two-wheeled nonholonomic vehicle is a problem deeply analyzed by several papers in the past (see references below) it is also possible to calculate the right parameters.

P. Muir and C.P. Neuman, "Kinematic Modeling of Wheeled Mobile Robots"

G. Campion, G. Bastin, B. Dandrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots"

Arturo Falck et al. has written an article where he shows how to calculate these values.

A robot which utilizes a PID algorithm can be found here (see Figure 19).

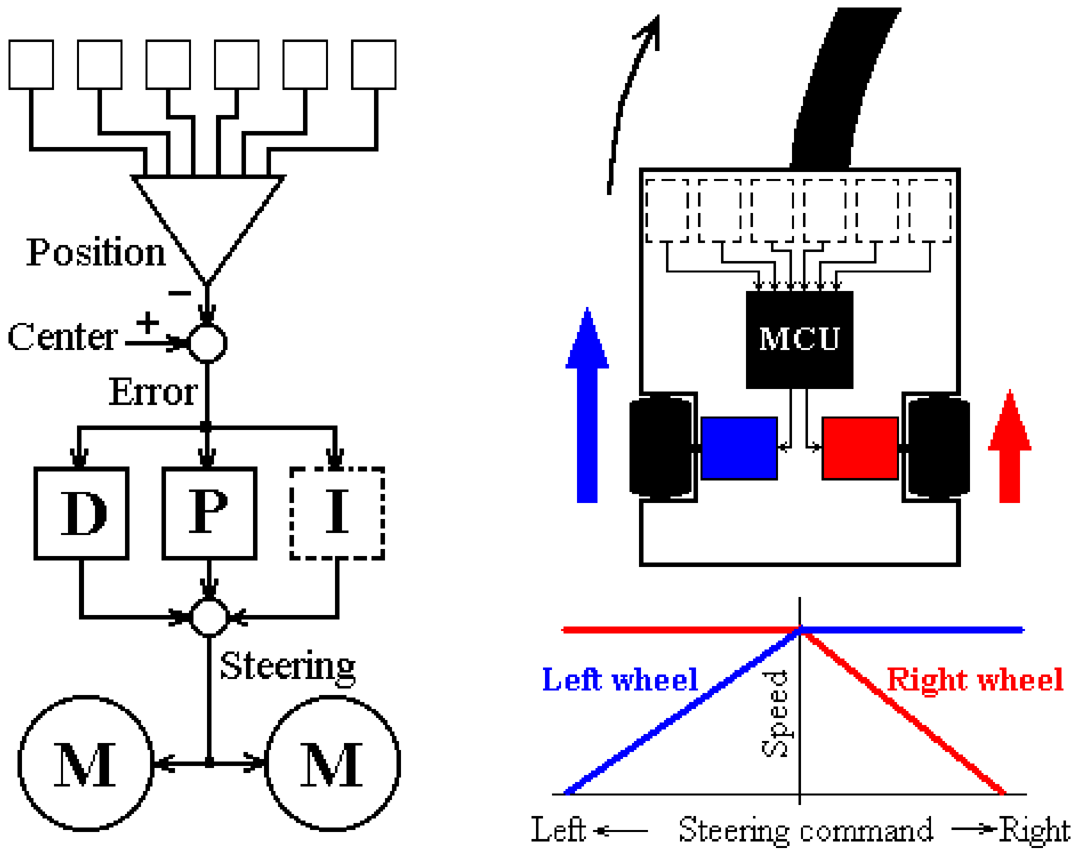


Figure 19: PID steering scheme by ChaN

4.3 Fuzzy based algorithms

Fuzzy logic is exemplified by the use of linguistic variables to represent part of the range a ordinary crisp variable may assume. These advance technique allow the control of complex systems. Once again there are several papers covering this topic:

Robert F. Reuss, Tsai-Ming Lee, "Fuzzy Logic Control in a Line Following Robot"

A. Saffiotti, "The Uses of Fuzzy Logic in Autonomous Robot Navigation: a catalogue raisonne (1997)"

4.4 Cerebellar Control

David Collins, Gordon Wyeth, "Cerebellar Control of a Line Following Robot"

Abstract: This paper proposes a robot control architecture inspired by the biology of the cerebellum. The cerebellum is thought to be the location of movement coordination in the body. It is hypothesized that through the use of some form of learned internal model, the cerebellum is able to overcome inherent sensory latency and coordinate fast, accurate movement without the need of a complex mathematical algorithm. The study presented in this paper attempts to encapsulate these attributes in an introductory experiment based around a line following robot. The robot learns to negotiate a circuit using a crude supervising module as the teacher, and eventually becomes more proficient at performing the task than the example on which it based its experience.

5 Future Prospects

5.1 Planned Applications

5.1.1 MCLU Robot

As mentioned above the Line Follower should be used as part of the microcontroller course at the Vienna University of Technology where students would have to programm the robot. Mario Grotschar is working on a system allowing students to program the robot from home while watching its behavior by a webcam. The robot itself is contacted by the wireless interface by a server.

5.1.2 Balancing Robot

Like Segway HT or nBot a self balancing robot should be build which utilizes the Line Follower's PCB. See Figure 20.

5.2 Possible Other Applications

Because of the openness of the design a great deal of other applications can be imagined.

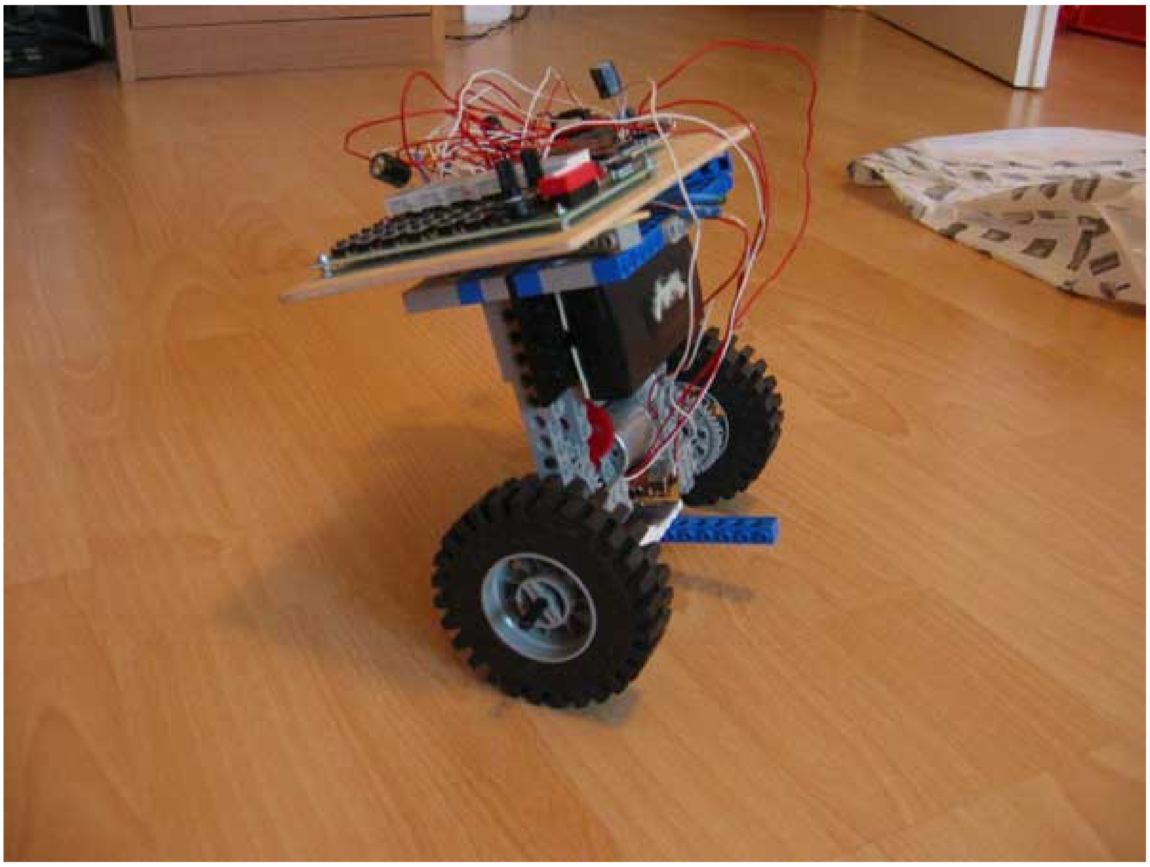


Figure 20: Balancing Robot Prototype

6 Final Notes

6.1 Lessons Learnt

This practical was very instructional. Nearly fifty percent of the prototyping effort were spent in the planning phase, which was a difficult job, because I have never done anything like this before. It took a long time to get an overview of all available components, to read the datasheets and compare them to each other. Many interdependencies had to be kept in mind while changing a parameter, for example the supply voltage. I learnt how to use Eagle for creating schematics and how to set up an AVR GCC environment under Windows XP. I mention this because it was a bit tricky to find the correct COM port settings under Windows.

After all it was great fun to play with Lego again. I think it is very well suited for rapid prototyping of small robots, especially because you can machine it very easily with a Dremel tool to accommodate your needs. To build the hardware took only 20% of the prototyping work, and the effort was focused mainly on the sensor development. Software implementation and testing took 30 percent of the prototyping work. It was a great satisfaction to see that the robot actually works very well, and it is great fun to play with it. I think with the right software it should be possible to double its speed, so I still work on it when I'm in the mood.

The second big challenge was to create the PCB, because it was my very first Eagle Project, and the design is rather complex. Especially the routing was very difficult, due to the lack of space. The whole PCB drawing took me 3 months, but now I'm feeling very comfortable about working in Eagle. Many thanks to Günther Gridling who helped me a lot.

The last obstacle was creating this document, which was my $\LaTeX$premiere. This took about 2 weeks.

6.2 Problems & Pitfalls

As mentioned before one of the biggest challenge was to find the correct parts. It was also interesting to solve the overheating problem of the L293D motordriver. The self-made cooler attached to the IC made a huge difference. When the robot gets more and more slowly, and you blow air over the cooling fins it gets faster immediately, which is really exiting to see.

6.3 Conclusion

I had great fun and I learnt a lot, so I can only recommend building a robot to every student interested!

If you have any questions please feel free to write me an email: e0126310@stud4.tuwien.ac.at