



# AREXX ARDUINO ROBOT AAR

MANUAL: AAR-04



© AREXX - The Netherlands V062012

# Contents

1. PRODUCT DESCRIPTION AAR	
1.1 The ARDUINO Robotics Family	3
1.2 Specifications	3
2. ARDUINO General Description	5
3. AREXX ARDUINO ROBOT	10
3.1. Blockdiagram	10
3.2. AAR hardware	11
3.2. ARDUINO Software	12
4. The AREXX ARDUINO ROBOT (AAR)	13
4.1. Download and installation of the software	13
4.2. The Arduino language	13
4.3. Installation of a USB-driver	13
4.4. AAR Hardware	14
4.4.1. Installing the battery-compartment	14
4.5. ARDUINO Software	15
4.5.1. Programming with Arduino Programs.	15
4.5.2. Selecting an Arduino Program	15
4.5.3. Selecting the correct COM-port	16
4.5.4. Program transfers to the Arduino Robot	17
6. Background-information to the H-Bridge circuits	18
7. Odometry	21
8. Programming a Boot-loader	24
<b>9. APPENDIX</b>	<b>25</b>
<b>9.1 Parts List</b>	<b>26</b>
<b>9.2 Main Board - Top View</b>	<b>28</b>
<b>9.3 Main Board - Bottom View</b>	<b>29</b>
<b>9.4 Schematics AAR</b>	<b>30</b>

## NOTICE!

AAR is a trademarks of AREXX, The Netherlands and JAMA, Taiwan.  
AREXX and JAMA are registered trademarks

## All rights reserved.

Reprinting any of this instruction manual without our permission is prohibited.  
The specifications, form, and contents of this product are subject to change without prior



Manufacturer:  
AREXX Engineering  
JAMA Oriental



European Importer:  
AREXX Engineering  
ZWOLLE Holland

Technical help:

**WWW.AREXX.COM**  
**WWW.ROBOTERNETZ.DE**

© AREXX Holland and JAMA Taiwan  
© English translation: AREXX - The Netherlands

# **1. PRODUCT DESCRIPTION AAR**

## **1.1. The ARDUINO Robotics Family**

Arduino is an open source-Platform for developing of electronic prototypes, which provides us with a microcontroller including all peripheral interfaces and the required software.

The Arduino-concept has been designed to learn modern electronics for robotics, software control and sensors in the simplest possible way.

As a successor for the ASURO-robot, which has been programmed in C-language we now designed the AREXX Arduino robot. The new robot resembles its predecessor ASURO, but in combination with an „open source“- programming language Arduino programming the system will be much easier.

## **1.2. Specifications:**

Motors	2 DC-motors (3 Volt)
Processor-type	ATmega328P
Programming language	ARDUINO
Supply voltage	4 x AAA-type batteries 4,8 - 6 Volts
Supply current	Min. 10 mA Max. 600 mA
Communication	USB-plug
Extensions	ASURO-extensions are compatible
Height	40 mm
Width	120 mm
Depth	180 mm

## **1.3. Precautions**

- 1. Attention!** You must read this manual before supplying power to any of the terminals! Incorrect connections may damage the hardware.
- 2. Attention!** Please check the pin function diagram carefullyBe careful in wiring the circuitry. Incorrect connections may damage the modules. Respect the correct power supply's polarity. A reversed power supply may damage the hardware.
- 3. Attention!** Don't use power supply with voltages beyond the rated voltages! Use stabilized and filtered power supplies to avoid voltage and spikes.
- 4. Attention!** The board does not provide any waterproof or wet proof protection. Please use and save the system in dry environment.
- 5. Attention!** Avoid short circuits at any metallic surface and do not stress the printed circuit board or the plugs by excessive forces or weights.
- 6. Attention!** Be careful to avoid ESD (see prevention measures, precautions and descriptions at Wikipedia's Electro-Statical Discharges).

## **1.4. General Precautions**

- \* When you open the parts the return right will be disposed
- \* Read before you start assembly the instruction manual
- \* Be careful with tools
- \* Keep this product out of reach of children and do not build this kit when children are in the neighbourhood, the tools and parts are dangerous for children
- \* Check the polarity of the batteries
- \* Keep the batteries dry, when the ASURO gets wet remove the batteries and let the AAR dry for some time
- \* Remove the batteries when you are not using the robot for a longer period

## 2. ARDUINO General Description

### 2.1. Who or what is ARDUINO?

Arduino is an open source- single board microcontroller, which provides an easy access to programming, microcontrollers and project platforms for interactive objects for artists, designer, hobbyists and others.

The Arduino-platform has been based on an Atmel's ATmega168 or ATmega328 microcontroller. The system provides users with digital I/O-ports and analog input channels, which allow the Arduino-system to receive and respond to signals from the environment.

The market supplies us with several Arduino-boards such as Arduino Uno, Arduino LilyPad and Arduino Mega 2560. Each Arduino-board has been designed for specified purposes and users obviously may choose an ideal Arduino-assembly for almost any project.

For example input signals may be delivered by switches, light sensors, speed and acceleration sensors, proximity sensors and temperature sensors. Additionally commands will be allowed from any web-sources. Output-signals will be used to control motors, pumps and screen displays.

The system has been equipped with a compiler for a standardized programming language and a boot-loader. The programming language has been based on Wiring- language, which corresponds to C++.

Originally the Arduino project started 2005 in Ivrea, Italy. The concept aimed to support students in projects, in which the prototyping should be cheaper and more efficient as in most standard methods. The developer group under Massimo Banzi and David Cuartielles decided to name the project after a historical character named 'Arduin of Ivrea'. "Arduino" is the Italian version of the name, meaning "strong friend".

The English version of the name is "Hardwin".

## **2.2 Microcontrollers!**

### **2.2.1. Applications**

A microcontroller (sometimes abbreviated  $\mu$ C, uC or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory and a small amount of data memory (RAM) is also often included on chip.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, and toys. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes.

A typical home in a developed country is likely to have four general-purpose microprocessors and three dozen microcontrollers. A typical mid-range automobile has as many as 30 or more microcontrollers. They can also be found in many electrical device such as washing machines, microwave ovens, and telephones.

## **2.3. Power Consumption and Speed**

Some microcontrollers may operate at clock rate frequencies as low as 4 kHz, for low power consumption (milliwatts or micro-watts). They will generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just nanowatts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles, where they may need to act more like a digital signal processor (DSP), with higher clock speeds and power consumption.

The Arduino system applies a powerful Atmel ATmega328P single-chip, providing an 8-bit microcontroller at 16 MHz with 32K bytes In-system programmable flash. The power supply voltage has been designed quite versatile in the range DC7-12V, providing stabilized and protected operating conditions for the chip and isolated power lines up to 2A for motor circuitry.

## **2.4 Microcontroller Programs**

Microcontroller programs must fit in the available on-chip program memory, since it would be costly to provide a system with external, expandable, memory. Compilers and assemblers are used to convert high-level language and assembler language codes into a compact machine code for storage in the microcontroller's memory. Depending on the device, the program memory may be permanent, read-only memory that can only be programmed at the factory, or program memory may be field-alterable flash or erasable read-only memory.

Microcontrollers were originally programmed only in assembly language, but various high-level programming languages are now also in common use to target microcontrollers. These languages are either designed specially for the purpose, or versions of general purpose languages such as the C programming language. Microcontroller vendors often make tools freely available to make it easier to adopt their hardware.

The Arduino system provides us with approximately 32K bytes of flash-memory for sketches programs, which may be programmed in C programming language.

## 2.5. Interface Architecture

Microcontrollers usually contain from several to dozens of general purpose input/output pins (GPIO). GPIO pins are software configurable to either an input or an output state. When GPIO pins are configured to an input state, they are often used to read sensors or external signals. Configured to the output state, GPIO pins can drive external devices such as LEDs or motors.

Many embedded systems need to read sensors that produce analog signals. This is the purpose of the analog-to-digital converter (ADC). Since processors are built to interpret and process digital data, i.e. 1s and 0s, they are not able to do anything with the analog signals that may be sent to it by a device. So the analog to digital converter is used to convert the incoming data into a form that the processor can recognize. A less common feature on some microcontrollers is a digital-to-analog converter (DAC) that allows the processor to output analog signals or voltage levels.

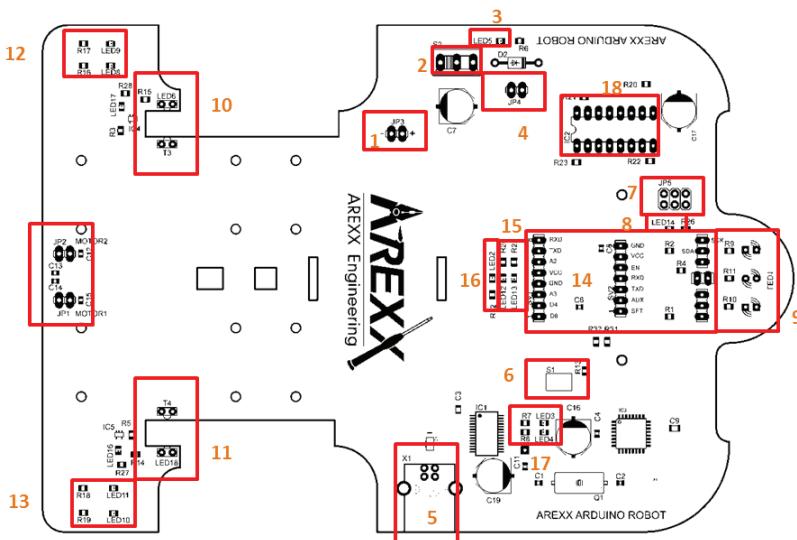
In addition to the converters, many embedded microprocessors include a variety of timers as well. One of the most common types of timers is the Programmable Interval Timer (PIT). A PIT just counts down from some value to zero. Once it reaches zero, it sends an interrupt to the processor indicating that it has finished counting. This is useful for devices such as thermostats, which periodically test the temperature around them to see if they need to turn the air conditioner on, the heater on, etc.

Universal Asynchronous Receiver/Transmitter (UART) block makes it possible to receive and transmit data over a serial line with very little load on the CPU. Dedicated on-chip hardware also often includes capabilities to communicate with other devices (chips) in digital formats such as I2C and Serial Peripheral Interface (SPI).

The Arduino system provides us with 14 digital I/O-lines, 7 analog I/O-lines.

### 3. AREXX ARDUINO ROBOT

Fig. :AAR PCB



#### 3.1 ARDUINO ROBOT

#### Block diagram

1. Connector plug for the battery compartment. (Be careful to check for the correct polarity!)
2. On/Off-Switch for the Robot.
3. Status-LED: signaling that the robot is being supplied from the power supply.
4. In case you are using rechargeable batteries you may interconnect this dual plug, which will supply the robot with the correct supply voltage
5. USB-connector to program the robot with the help of the Arduino-Software.
6. Reset-button: to be used to manually reset the robot.
7. ISP-connector, which may allow you to install another bootloader program.
8. LED 14: this LED provides free access for all programming and will blink if the bootloader is (re-)started.
9. Line-follower: This module provides free access for programming and allows the robot to follow lines.
10. Wheel-sensor left: this module generates pulses proportionally to the rotation of the left wheel.
11. Wheel-sensor right: this module generates pulses proportionally to the rotation of the right wheel.
12. Status LEDs for the left-sided motor: These LEDs indicate the motor's forward, respectively backward rotation.
13. Status LEDs for the right-sided motor: These LEDs indicate the motor's forward, respectively backward rotation.
14. Connector for the extension board, in which for example an APC220 wireless module or a Snake Vision-module may be installed and connected to the Arduino-System.
15. Status LEDs for the RS232 communication interface.
16. Status LED 2: freely accessible LED for programming.
17. Status LEDs for USB data-communication.
18. Motor-controller

## **3.2 Background-information for the AAR**

The front-side provides a USB-interface equipped with an FT232 IC. This chip transforms the USB-signal into a RS232 UART-signal, which may be processed by the ATMEGA328P Processor (located at right side of the front position).

At the opposite side we positioned the ON/OFF-switch including a JP3-connector for the supply-connection and the motor-controller IC2. The back-side of the printed circuit board (PCB) has been chosen to locate both engines and the wheel-sensors.

The wheel-sensors are using photo-eyes. The cogwheels have been equipped with four holes in a 90°-position pattern. As soon as the light passes a hole and hits a sensor the wheel sensor will send a trigger-pulse for this corresponding wheel to the processor. Additionally the electronic circuits switch on LED16 respectively LED17. The trigger pulses allow us to have an accurate overview of the wheel-speed for each of the rear wheels.

At the front-side we located the connectors for extension boards and at the bottom-side of the PCB we will find the sensors for the line-follower circuit.

The line-follower uses an LED to send a beam of light to the bottom area. Alongside of the LED two infrared sensors have been positioned to monitor the reflected light from the bottom. Additionally the PCB provides us with the other components (LEDs, resistors and capacitors) to complete the line-follower to a working module.

The robot uses an Arduino-board, which may be compared to the Arduino Duemilanove board. The ATMEGA328P micro-controller is the system's core, which provides us with 14 digital I/O-ports, in which six ports are configurable as Pulse-Width-Modulated (PWM-) output channels. Additionally the robot has been configured with 6 analog input channels, a 16MHz crystal oscillator and a USB- connector for programming and control. The list may be completed with an ISP-connector, enabling experienced hobbyists to program their own boot-loader program.

The robot has been designed for a 5V-supply voltage and may also be satisfied with the supply current from the USB-plug. This option is quite comfortable for testing and programming.

Rather comfortable in this robotic concept are the connector plugs, which allow you to insert your own extension modules or the AREXX-extension modules from the ASURO-series.

### **3.3 BACKGROUND-INFORMATION FOR THE ARDUINO SOFTWARE**

Arduino Software belongs to the Open Source-category and is universally available to all, including the source codes for the programming platform.

The Arduino programming platform has been equipped with a text-editor, a message window and a text-console. The programming platform may directly contact the AAR for communication and allows us to easily transfer programs into the processor.

Programs, which have been written in Arduino-language, are named „sketches”. A normal text-editor is used for developing and editing these programs. The “sketch”-files will be stored at your PC’s hard disc. Sketches are identifiable by their file-extension „.ino”.

Saving-actions the sketch-files are reported in the message-window, which also includes detected errors in the source-code. The right-sided bottom of the window displays the currently active Arduino-Board and the serial interface .

The basic Arduino-concept supplies us with libraries filled with extra functionality. A library defines a number of predefined functions, which for recurrent programming sections may be reused at no extra cost for development.

Basically an Arduino-program may be structured in three sections:

- structure,
- definitions (for variables respectively constants) and
- functions.

An Arduino-structure consists of a setup and a loop-function. The setup is used to initialize variables, pin-definitions („Pin-Modes“) and libraries-definitions.

The „Loop“-function will be repeated in an endless loop, which allows the program to react permanently ad lib, until the system is switched off.

The program uses „variable“-definitions to store and handle a program’s modifiable data whereas constants are used to define fixed values such as pin-definitions for input- or output-functionality and to define fixed voltage levels at pin-connections.

## **4. Getting Started**

### **4.1. Download and installation of Arduino's Software**

Install the Arduino software (version 1) from the CD we are sure this will work. Later you also can go to the ARDUINO website and download the latest version from this site.

***IMPORTANT:***

using different versions of the ARDUINO Software and different version of the application software may give some problems. Somtimes with a new ARDUINO software update you have to modify your application software otherwise it wil not work!

### **4.2. Arduino's language**

The grammar of Arduino's language has been documented in the official Arduino website. Learn to understand the specific language's characteristics to the level you need.

### **4.3 Installation of an USB-driver**

When you connect the board, Windows should initiate the driver installation process (if you haven't used the computer with an Arduino board before). On Windows Vista or higher, the driver should be automatically downloaded and installed.

Select the serial device of the Arduino board from the Tools > Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports).

To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.

## 4.4. AAR hardware

### 4.4.1. Installing the batteries

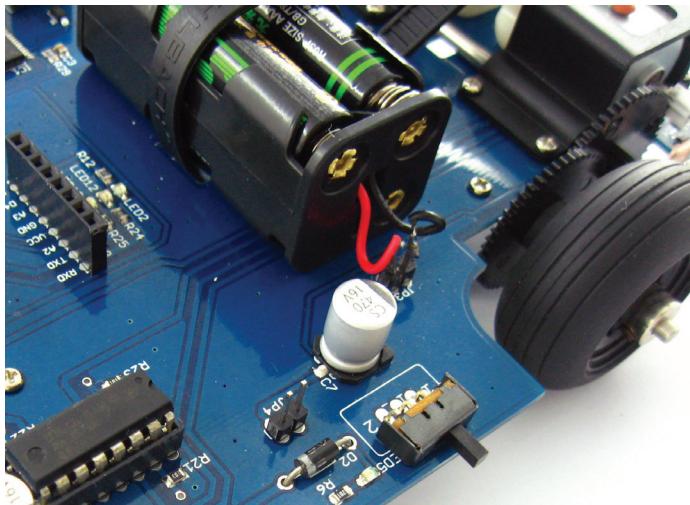
The robot has been designed for a power supply filled with four 1,5V AAA-cells. If you prefer to use rechargeable batteries the jumper JP4 should be installed as a bridge to prepare the system for a lower voltage of the rechargeable batteries (see fig. 1, number 4).

#### **ATTENTION!**

**Installing the connecting jumper JP4 will disable the polarity check using the rectifier diode. Errors in power connections with installed jumper JP4 might seriously damage the robot.**

Connect the battery compartment as shown in the figure (fig. 2)

**Fig. 2:**  
**Battery connection**



Now you may switch on the robot by activating the ON/OFF-switch. Located directly besides the switch the power LED (LED5) will be illuminated.

## 4.5 ARDUINO software

### 4.5.1 Programming the Robot with Arduino Programs.

Connect the robot by USB-cable to your PC.

As soon as the robot has been connected to an USB-port the Arduino-system does not really need an extra battery or other power supply. Instead the USB-connection to the PC will provide the required power supply.

#### **ATTENTION:**

**The robot will always be activated as soon as the system has been connected to the PC. The ON/OFF power switch and LED5 will only be active in the case of battery powered operation.**

Now you may open the Arduino Software (see fig. 3a).

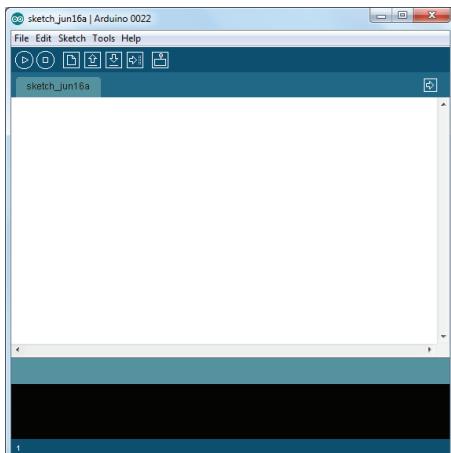


Fig. 3a Arduino software

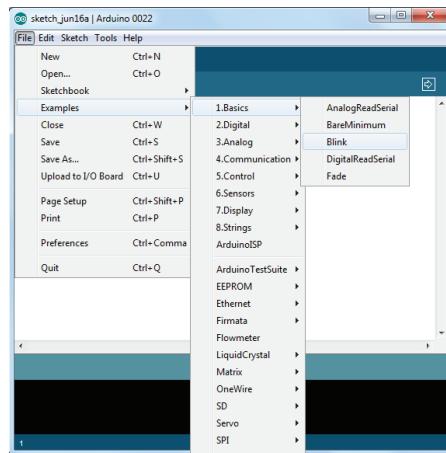


Fig. 3b Opening Blink program

### 4.5.2 Selecting an Arduino Program

We will start by loading a simple sample program named „blink“ into the robot. The program will command the robot to repeatedly flash LED1.

Load the program by searching and clicking the program in Arduino's software at the menu-entry

File>Examples>1. Basics>Blink (see fig. 3b), which will display the following messages at the platform's window (fig. 4a).

```

Blink
Turns on an LED on for one second, then off for one second, repeatedly.

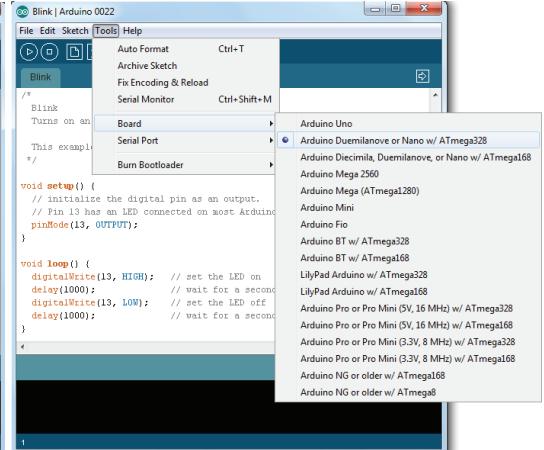
This example code is in the public domain.

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // set the LED off
  delay(1000);           // wait for a second
}

```

**Fig. 4a Program Blink**



**Fig. 4b Select Board**

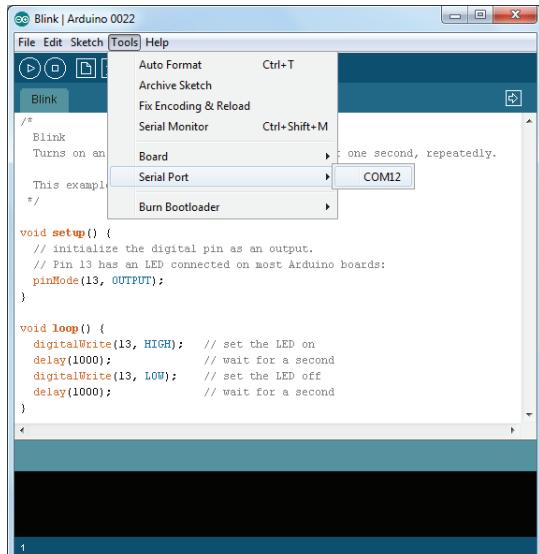
At this stage we will have to select the correct Arduino-board at the menu-entry **Tools>Board> Arduino Duemilanove or Nano w/Atmega328** (see fig. 4b)

#### 4.5.3 Select Compoort

The next step defines the correct COM-port for the Arduino-interface. The correct COM-interface (or COM-port) for the robot is COM 12.

In order to select the COM-interface please open the menu-entry: **Tools>Serial Port>COM 12.**

(see fig. 5)



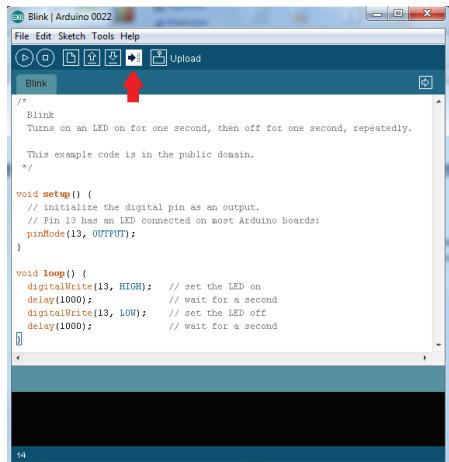
**Fig. 5 Selecting the correct Com-Port**

#### **4.5.4. Program transfers from the PC to the Arduino Robot**

Please click the button, which has been marked with a red arrow (or alternatively follow the menu-entry „File>Uploading to I/O board“) to transfer the selected program to the connected Arduino-robot (see fig. 6a).

The status window reports the compilation process of the program and as soon as the program successfully has completed the compilation the system will start the upload to the robot.

At the end of the upload the status window reports: „Done uploading“ (See fig. 6b).



**Fig. 6a Uploading software**

A screenshot of the Arduino IDE showing the same sketch after the upload process has completed. The status bar at the bottom of the window displays the message "Done uploading". Above this, the text "Binary sketch size: 1018 bytes (of a 30720 byte maximum)" is shown. The rest of the window content is identical to Fig. 6a, displaying the sketch code.

**Fig. 6b Ready with the upload**

***At this stage you might remove the USB-cable to disconnect the robot from the PC, connect the battery-compartment or power supply and start the robot.***

For further information and downloads we invite you to visit one of the forums at the websites:

**[www.arexx.com](http://www.arexx.com) --> Forum**  
**[www.roboternetz.de](http://www.roboternetz.de) --> Forum**

# 5. Background-information to the H-Bridge circuits

A H-bridge is an electronic circuit which allows us to reverse the polarity of a device (such as a DC-motor) by controlling four switches. These H-bridges will often be found in robotics to control a motor rotation in two opposite directions.

Modern systems use integrated circuits for motor control, but to learn the basic fundamentals and the dimensioning problems of power supplies it might be important to study an archaic circuit for motor controls.

## 5.1 A H-Bridge for 3 Volt Power supplies

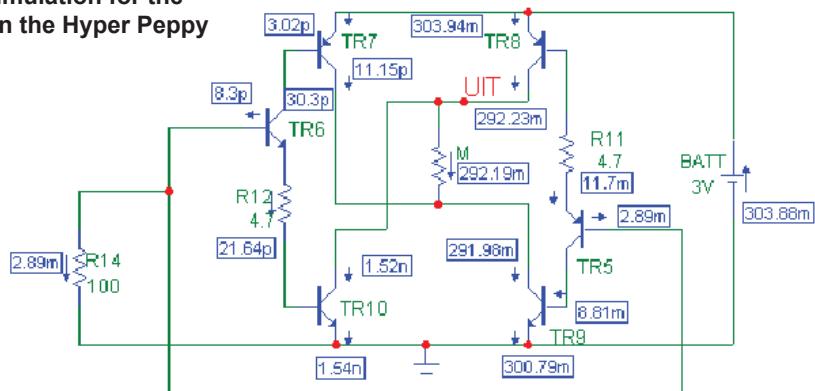
The driver circuit for the Hyper-Peppy robots contains two PNP-Transistors TR7 and TR8, respectively NPN-Transistors TR9 and TR10. In this design we always allow only two transistors to simultaneously conduct currents into motor M:

via TR7 and TR10 or alternatively  
via TR8 and TR9.

The (freely available test-version of the) Microcap simulator allows us to comfortably calculate the DC-simulation for the circuit and read the values from the schematic window:

|

Afb. 8: simulation for the H-bridge in the Hyper Peppy robot



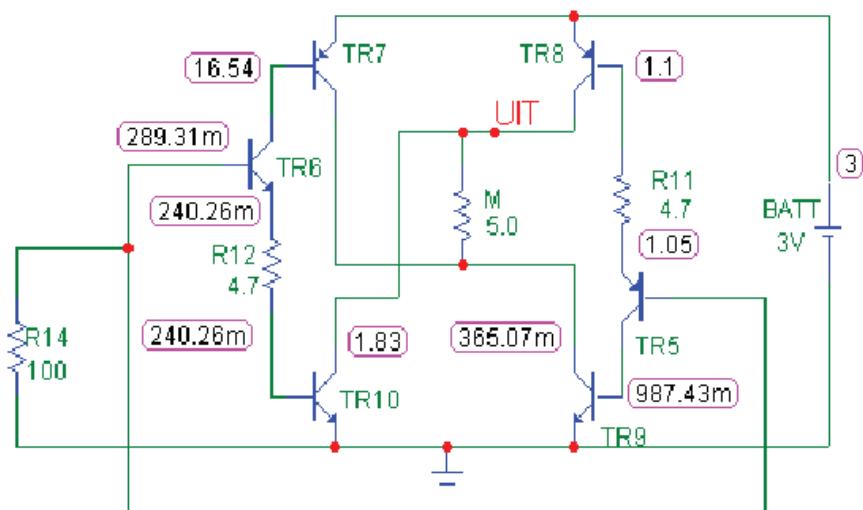
In the driver stage we may identify the DC-motor M. The preamplifier of the driver circuit is being simulated by resistor R14. This resistor will pull the base-ports of transistors TR6 and TR5 to 0V, which results in a condition in which only the right-sided branch is conducting a significant current.

Transistors TR8, TR5 and TR9 are conducting and the other transistors are blocked. As soon as we switch R14 to a positive voltage the right-sided branch will be blocked and the motor current will be reversed.

The Microcap Simulator allows us to calculate the currents for all components and read the values from the schematic window. The total supply current at 3V battery voltage will be circa 300mA.

The remarkable low supply voltage for this circuit depends on the combination of silicon PNP- and NPN- transistors, which both work with 0.7V knee voltages. The motor however has been designed between two collector ports, which in a saturation mode merely conduct at 0.3V. For the motor M these switches supply the motor with a respectable 1.5V. As calculated by Microcap the values may be read from fig. 9.

**Fig. 9: DC-settings for the H-Bridge  
in the Hyper Peppy Robot**



The 3V-power supply is an ideal condition for a robot with a battery-pack of only 2 cells. The PNP-transistors however cannot easily be integrated in an IC such as the L293D. An IC however has other advantages such as reliability, protection against bad circuitry and reduced PCB-area and low weight. For this reason we decided to use a L293D-chip with a dual H-bridge circuitry to simultaneously control two DC-motors.

## 5.2 A H-Bridge for 4,5 Volt

The L293D-chip (see fig. 10) allows us to control output-currents up to 600mA per channel (maximal: 1.2A peak currents). The power supply voltage of the drivers (VCC2) may vary between 4.5V and 36V, which promotes this L293D-Chip to a favorite circuit for DC-motor control.

The minimal power supply voltage (VCC2) however has been specified as 4.5V, which forces us to choose a minimum of 4 rechargeable batteries as a power supply. This investment increases the robot's total weight. It is the price we pay for modern electronic circuitry.

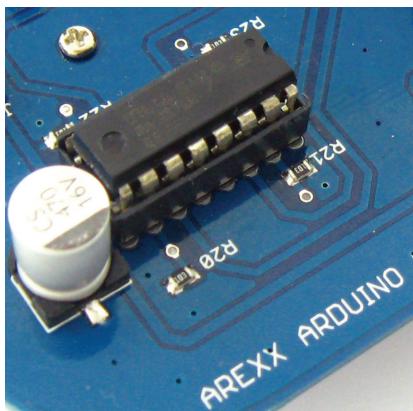


Fig. 10 H-Bridge circuitry with a L293D-Chip

## 6. Odometry

This chapter has been devoted to some interesting application-concepts for the AAR-robot. The ideas refer to studies and art-projects. Maybe developing such software will inspire us in programming micro-controllers.

### 6.1 Line-seekers, color-lovers and color-haters

Light-sensitive sensors will allow us to program robots to behave like line-seekers, color-lovers or color-haters. In the first of these examples the robot should be expected to follow a line in an 8-shaped curve, which in an endless loop will force it to run around eternally.

The second and third example teaches the robot to avoid red light sources and maybe feel attracted by green light simultaneously. These kind of behavior patterns already will be considered as practical survival patterns for simple living organisms such as worms.

### 6.2 Cowards and music-lovers

Quite interesting is also a behavior pattern which depends on environmental noise. A jumpy robot equipped with a sensitive microphone may be taught to avoid heavy base music but simultaneously be attracted by high-toned flute-music. The attraction of high frequencies may even override the fear for heavy low frequencies. This way the robot may be forced to feel attracted by the source of the high flute-music in spite of the heavy-metall music.

Behavior patterns which depend on high and low frequencies, light and colors merely need a few sensors, two frequency filters and a couple of light-dependent sensors, equipped with color filters.

### **6.3 Complex Line-seekers**

Line-seeking and line-following robots usually will need a light source such as a LED and two or more light sensors. These devices allow the system to identify a line and follow the track. Initially the robot may need a special search pattern routine to find a line. The pattern may consist of a strategy to follow a spiral pattern with a gradually increasing radius from the starting point. This pattern is to be followed until the sensor detects a specific line pattern and starts following this line.

Developing such programs, which sufficiently solves the problem of searching any kind of line patterns, already belongs to the sophisticated categories of software.

#### **6.3.1 Complex behavior patterns (as a programming task)**

The project may be complicated by specifying the search strategy to work in a complicated pattern of colored areas and lines and to find any red line which will lead the robot to a safe and dark "garage".

As soon as the noise pattern has been absent for a while the robot may leave its "garage" and start searching a green line, which may lead the device to another "garage" with bright, green light, in which the robot will feel comfortable – even in an environment with heavy music.

As soon as the music however does contain a flute's high frequencies the robot will feel uncomfortable. It will leave its carport to find the red lines, which will bring it back home to its dark "garage".

Experienced programmers are aware of the complexity and design requirements of a program which needs to serve the problems of a line-follower with several differentiated color and sound dependencies. Programmers will have to design a program with a hierarchical set of numerous functions. The modular, structured concept will allow us to write a complex, but reliable software, which is performing the specified task.

The software's complexity may inspire the programmers to feel admiration for the tiny living organisms, which are combining these behavior patterns with a regular search for food and a successful procreation strategy. It's the great effort of nature's life in perfecting behavior patterns again and again – to keep life alive....

## **7. Programming a Boot-loader**

### **ATTENTION!**

**The described road-map in this chapter requires  
programming experience!**

You will be able to upload an Arduino-bootloader into the micro-controller: for instance with the help of STK500.

In order to transfer any program, which has been written in Arduino language, into the Atmega micro-controller the Atmega-processor will have to be equipped with a special Arduino-boot-loader. This boot-loader will be needed to take care of the correct location for the coded characters inside the Atmega's memory.

In order to install the boot-loader we will need the following components:

- \* an AVR Programming Board (for instance the STK500 Board)
- \* a 12 Volt power supply
- \* the AAR-Robot including an on-board ISP-connector (fig. 7)
- \* a PC provided with a physical COM-Port (preferably NOT a USB-RS232 converter, which might result in risks by timing-errors)

Please install (respectively update) the current version of the Arduino-software, which will be found at the website [www.arduino.cc](http://www.arduino.cc). The downloaded file probably will be delivered as a .ZIP- or .RAR-type. Unpack these files and store the contents at your hard disc.

Please apply e.g. WINAVR to transfer the Arduino boot-loader to the robot.



**Fig. 7: ISP connector**

### **Attention!**

The ARDUINO software belongs to the freeware category and from time to time it may happen that Arduino software and Arduino boot-loaders do not properly cooperate!

If you experience such or similar problems you are invited to visit the various Arduino websites respectively forums

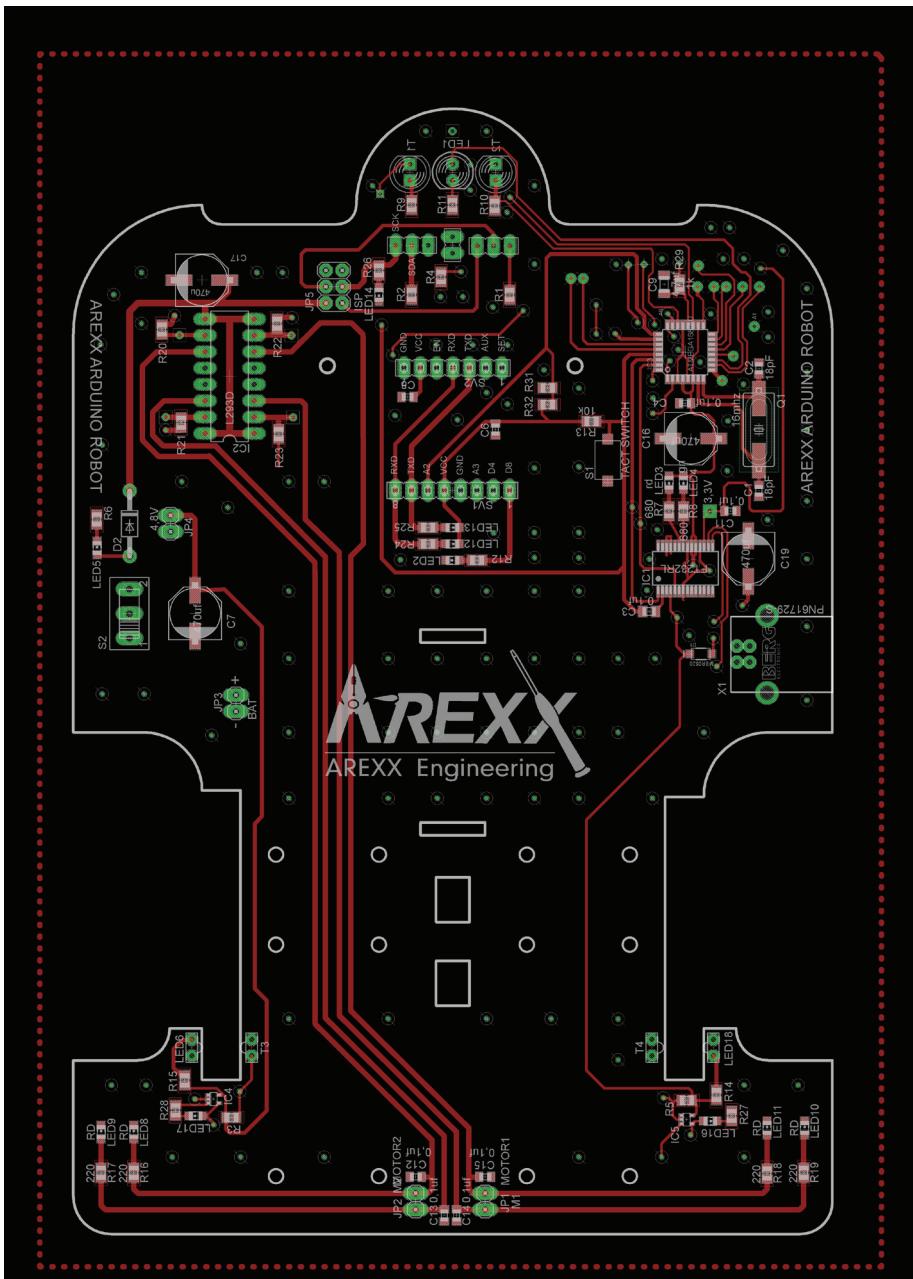
# **APENDIX**

# Partlist

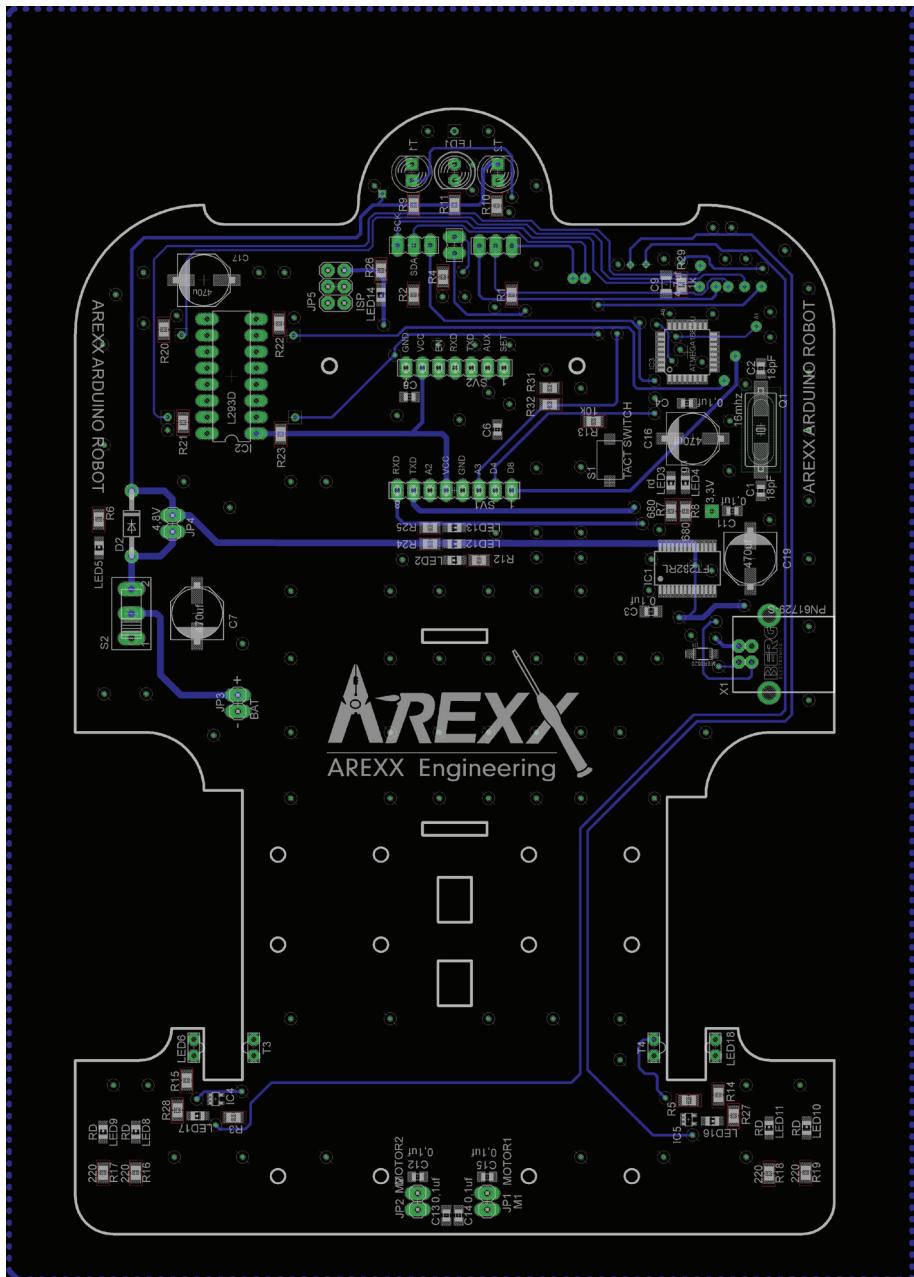
Part	Value	Package
C1	18pF	0805
C2	18pF	0805
C3	0.1uF	C0805K
C4	0,1uF	0805
C6	0,1uF	0805
C7	470uF	CPOL-USF
C8	0,1uF	0805
C9	4,7uF	1206
C11	0,1uF	0805
C12	0,1uF	0805
C13	0,1uF	0805
C14	0,1uF	0805
C15	0,1uF	0805
C16	470uF	CPOL-USF
C17	470uF	CPOL-USF
C19	470uF	CPOL-USF
D1	MBR0520	SOD-123
D2	1N4001	DO41-10
IC1	FT232RL	SSOP28
IC2	L293D	DIL16
IC3	ATMEGA168-AU	ATMEGA168-AU
IC4	74AHC1G14DCK	74AHC1G14DCK
IC5	74AHC1G14DCK	74AHC1G14DCK
JP1	M1	1X02
JP2	M2	1X02
JP3	BAT	1X02
JP4	4,8V	1X02
JP5	ISP	2X03
SV2	fem header	FE07-1
T1	SFH300	LED5MM
T2	SFH300	LED5MM
T3	LPT80A	LPT80A
T4	LPT80A	LPT80A
U\$1	3,3V	PIN-T
U\$2	FE03-1	FE03-1
U\$3	FE03-1	FE03-1
U\$4	FE02-1	FE02-1
X1	PN61729-S	PN61729-S
LED1	Rd	LED5MM
LED2	Bl	LEDCHIP-LED0805
LED3	Rd	LEDCHIP-LED0805
LED4	Gn	LEDCHIP-LED0805
LED5	Bl	LEDCHIP-LED0805
LED6	Rd	LEDIRL80A

Part	Value	Package
LED8	Rd	LEDCHIP-LED0805
LED9	Rd	LEDCHIP-LED0805
LED10	Rd	LEDCHIP-LED0805
LED11	Rd	LEDCHIP-LED0805
LED12	Gn	LEDCHIP-LED0805
LED13	Rd	LEDCHIP-LED0805
LED14	Bl	LEDCHIP-LED0805
LED16	Rd	LEDCHIP-LED0805
LED17	Rd	LEDCHIP-LED0805
LED18	Rd	LEDIRL80A
Q1	16MHz	CRYSTALHC49UP
R1	20k	R-US_R0805
R2	20k	R-US_R0805
R3	1k5	R-US_R0805
R4	220	R-US_R0805
R5	1k5	R-US_R0805
R6	1k	R-US_R0805
R7	680	R-US_R0805
R8	680	R-US_R0805
R9	20k	R-US_R0805
R10	20k	R-US_R0805
R11	220	R-US_R0805
R12	220	R-US_R0805
R13	10k	R-US_R0805
R14	220	R-US_R0805
R15	220	R-US_R0805
R16	220	R-US_R0805
R17	220	R-US_R0805
R18	220	R-US_R0805
R19	220	R-US_R0805
R20	10k	R-US_R0805
R21	10k	R-US_R0805
R22	10k	R-US_R0805
R23	10k	R-US_R0805
R24	220	R-US_R0805
R25	220	R-US_R0805
R26	220	R-US_R0805
R27	220	R-US_R0805
R28	220	R-US_R0805
R29/C3	0.1uF	C0805
R31	10k	R-US_R0805
R32	12k	R-US_R0805
S1	TACT SWITCH	TACT_SWITCH
S2	255SB	255SB
SV1	fem header	FE08-1

## A. MAIN PCB TOP



## B. MAIN PCB BOTTOM



## C. CIRCUIT AAR

