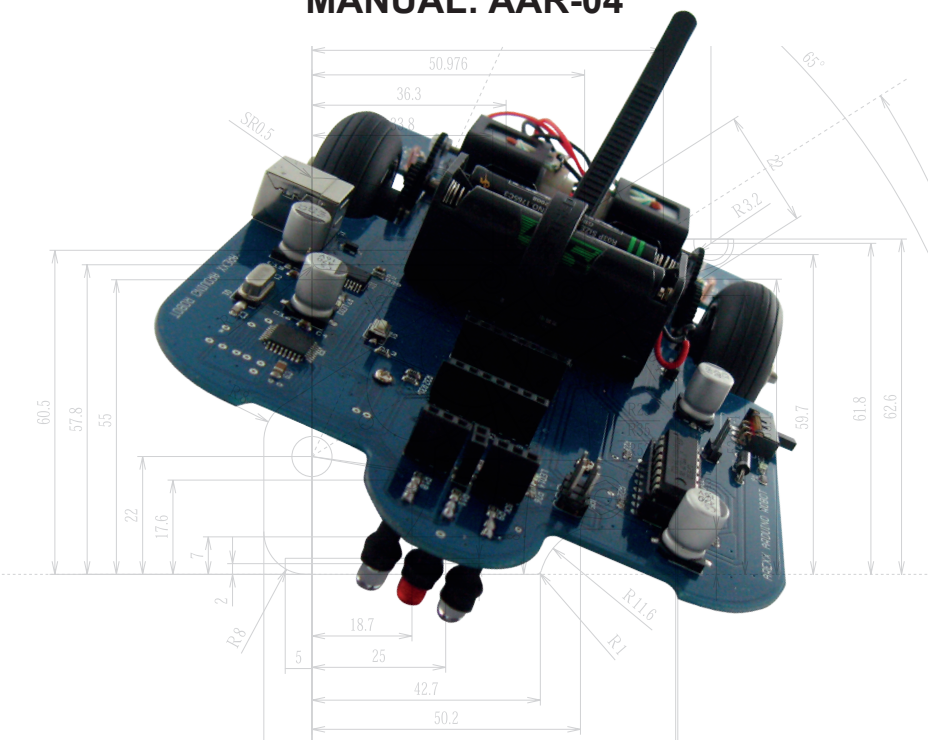
AREXX Engineering

ARDUINO ROBOT

AAR

MANUAL: AAR-04



Contents

Manufacturer:
AREXX Engineering
Zwolle, The Netherlands

AREXX Engineering

Technical help:
www.arexx.com
www.roboternetz.de

AREXX Engineering

Impressum

©2013 AREXX Engineering
Nervistraat 16
8013 RS Zwolle
The Netherlands

Tel.: +31 (0) 38 454 2028
Fax.: +31 (0) 38 452 4482

E-Mail: Info@arexx.nl

Information about limited warranty and responsibility

The warranty granted by AREXX Engineering is limited to the replacement or repair of the Robot and its accessories within the legal warranty period if the default has arisen from production errors such as mechanical damage or missing or wrong assembly of electronic components, except for all components that are connected via plugs/sockets.
The warranty does not apply directly or indirectly to damages due to the use of the robot. This excludes claims that fall under the legal prescription of product responsibility.

The warranty does not apply in case of irreversible changes (such as soldering of other components, drilling of holes, etc.) to the Robot or its accessories or if the Robot is damaged due to the disrespect of this manual!

The warranty is not applicable in case of disrespect of this manual! In addition, AREXX Engineering is not responsible for damages of all kinds resulting from the disrespect of this manual! Please adhere above all to the „Safety recommendations" in the Robot Arm manual

Please note the relevant license agreements on the CD-ROM!

IMPORTANT
Prior to using this robot arm for the first time, please read this manual thoroughly up to the end! They explain the correct use and informs you about potential dangers! Moreover it contains important information that might not be obvious for all users.

Important safety recommendation

This module is equipped with highly sensitive components. Electronic components are very sensitive to static electricity discharge. Only touch the module by the edges and avoid direct contact with the components on the circuit board. Please do never overload the motors.

Symbols

This manual provides the following symbols:

	*The "Attention!" Symbol is used to mark important details. Neglecting these precautions may damage or destroy the robot and/or additional components. You may risk your own health or the health of other persons!*
	*The "Information" Symbol is used to mark useful tips and tricks or background information. In this case the information is to be considered as "**useful, but not necessary**".*

Safety recommendations

- Check the polarity of the batteries or power supply.
- Keep all products dry, when the product gets wet remove the batteries or power directly.
- Remove the batteries or power when you are not using the product for a longer period.
- Before taking the module into operation, always check it and its cables for damage.
- If you have reason to believe that the device can no longer be operated safely, disconnect it immediately and make sure it is not unintentionally operated.
- Consult an expert if you are unsure as to the function, safety or connection of the module.
- Do not operate the module in rooms or area's under unfavourable conditions.
- Do not overload the motors.
- This module is equipped with highly sensitive components. Electronic components are very sensitive to static electricity discharge. Only touch the module by the edges and avoid direct contact with the components on the circuit board.

Normal use

This product was developed as an experimental platform for all persons which are interested in robotics. Main goal is to learn how you can program the device in ARDUINO. This product is not a toy; it is not suitable for children under 14 years of age!

This robot is also not an industrial robot arm, with industrial specifications and preformance!

It may only be used indoors. The product must not get damp or wet. Also be carefull with condence when you take it from a cold to a warm room. Give it time to adapt to the new conditions before you use it.

Any use other than that described above can lead to damage to the product and may involve additional risks such as short circuits, fire, electrical shock, etc.

Please read all the safety instructions of this manual.

1. PRODUCT DESCRIPTION AAR

1.1. The ARDUINO Robotics Family

Arduino is an open sourceplatform for developing electronic prototypes. It provides us with a microcontroller including all peripheral interfaces and the required software.

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

As a successor of the ASURO robot, which can be programmed in C-language, we now designed the Arduino robot. The new robot ressembles its predecessor ASURO, but in combination with an „open source" programming language, programming the Arduino system will be much easier.

1.2. Specifications:

Motors	2 DC-motors (3 Volt)
Processor-type	ATmega328P-AU
Programming language	ARDUINO
Supply voltage	4 x AAA-type batteries 4,8 - 5,5 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

Warnings

* The right of return does not apply after opening the plastic bags containing parts and components.
* Read the manual thoroughly prior to assembling the unit.
* Be careful when handling tools.
* Do not assemble the robot in presence of small children. They can get hurt with the tools or swallow small components and parts.
* Check the correct polarity of the batteries, do not mix different type of batteries.
* Make sure that batteries and holder always remain dry. If the ROBOT gets wet, remove the batteries and dry all parts as thoroughly as possible.
* Remove the batteries if the ROBOT will not be used for more than one week.

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
   carefully. Be 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).

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, designers, 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 enables 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. Users obviously can 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. Outputsignals 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 µC, 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 devices 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 a permanent, read-only memory that can only be programmed at the factory, or the program memory may be a 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.

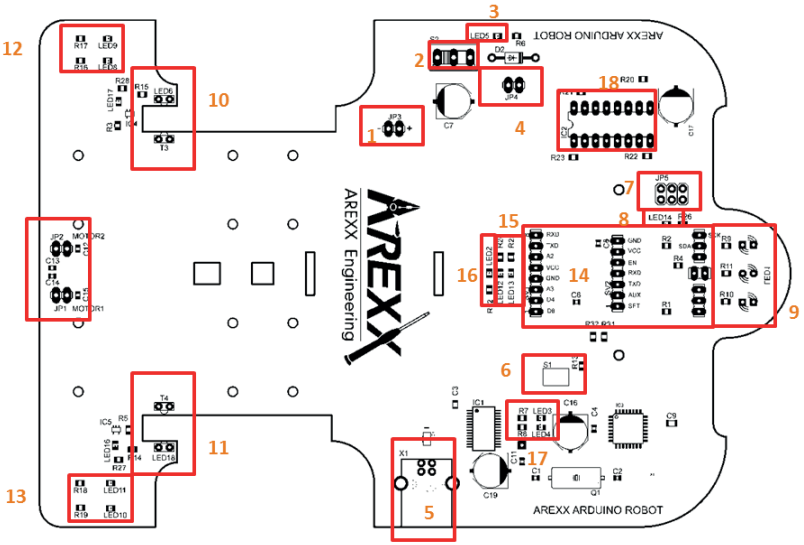
In addition to the converters, many embedded microprocessors include a variety of timers as well. One of the most common types of timers is the Programmable Interval Timer (PIT). A PIT just counts down from some value to zero. Once it reaches zero, it sends an interrupt to the processor indicating that 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 and 7 analog I/O-lines.

3. ARDUINO ROBOT

Fig.: AAR PCB



3.1 ARDUINO ROBOT LEDS

LED 1.	Red	Line sensor (bottom)
LED 2.	Blue	Free programmable
LED 3.	Red	USB DATA
LED 4.	Green	USB DATA
LED 5.	Blue	Power LED
LED 6.	IR	Infra Red LED left wheel sensor
LED 7.	NA	NA
LED 8.	Red	Motor indicator left forward
LED 9.	Red	Motor indicator left backward
LED 10.	Red	Motor indicator right backward
LED 11.	Red	Motor indicator right forward
LED 12.	Green	TX serial
LED 13.	Red	RX serial
LED 14.	Blue	Free programmable
LED 15.	NA	NA
LED 16.	Red	Right wheel sensor pulse
LED 17.	Red	Left wheel sensor pulse
LED 18.	IR	Infra Red right wheel sensor

LED 14 Blinks with only the bootloader
LED 14 is OFF without bootloader
LED 14 blinks shortly after power on, when a programm is loaded.

3.2 ARDUINO ROBOT PCB layout

See numbers in fig.: AAR PCB

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 connect 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: enables you to install another bootloader program.
8. LED 14: provides free access for all programming and will blink if the bootloader is (re-)started.
9. Linefollower: 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

On the CD we included the AAR selftest software AAR_SelftestV3.
With this program you can test all the basic functions of the AAR robot.

Please use the serial monitor in your Arduino Software, do not forget to install the libraries see page 22.

3.3 Background information for the AAR

The frontside provides a USB-interface equipped with a 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 backside 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 cog wheels 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 frontside we located the connectors for extension boards and at the bottomside of the PCB we will find the sensors for the line follower circuit.

The line follower uses a 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 into 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 bootloader 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 extension modules of the ASURO-series.

3.4 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 of 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 functionaility. 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 library 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 versions of the application software may give some problems. Sometimes, 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 a 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 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 below (fig. 2)

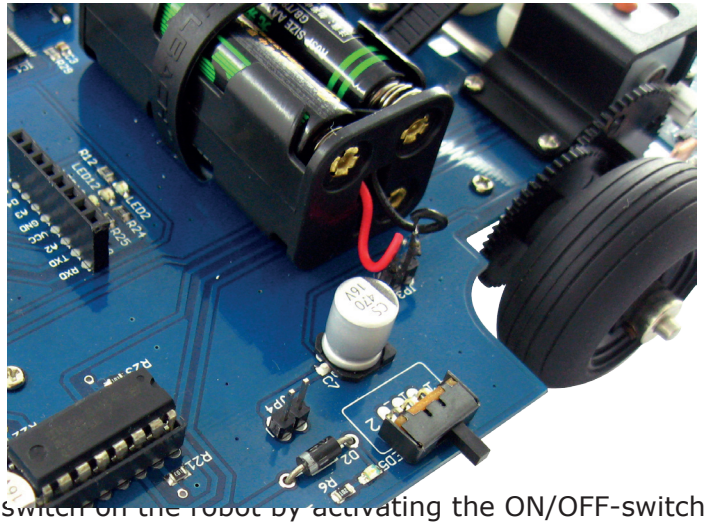


Fig. 2: Battery connection

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 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).

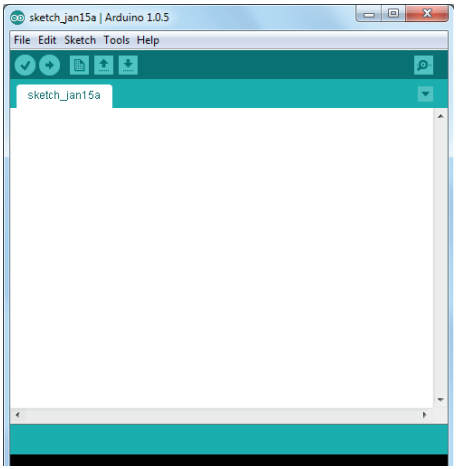


Fig. 3a Arduino software

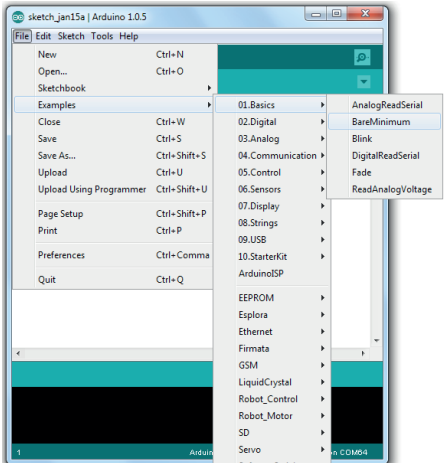


Fig. 3b Opening blink program

4.6 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).

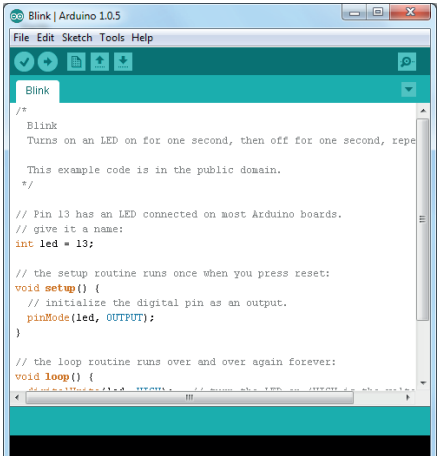


Fig. 4a Program Blink

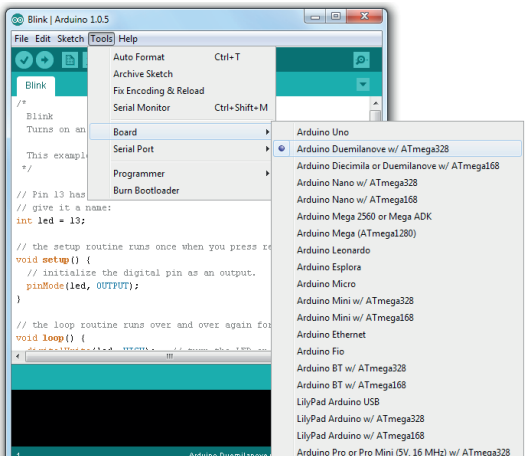


Fig. 4b Select Board

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

4.7 Select Comport

The next step defines the correct COM-port for the Arduino interface. The correct COM-interface (or COM-port) for the robot is normally higher as COM 3 (Com 1 and Com 3 are already in use.

In order to select the COM-interface please open the menu entry: **Tools>Serial Port>COM** x. (see fig. 5)

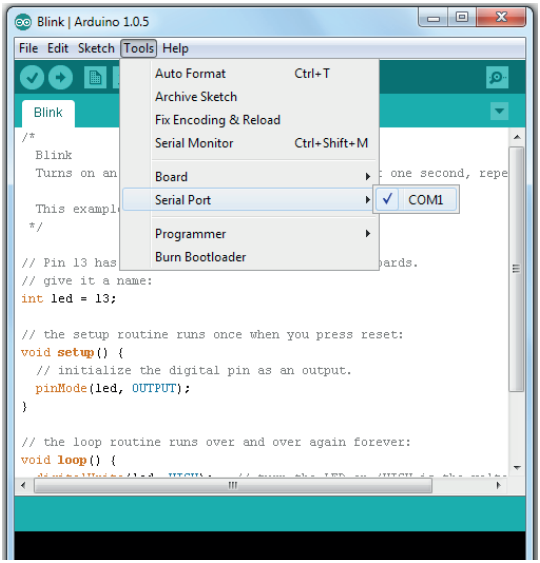


Fig. 5 Selecting the correct Com-Port

4.8 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), also LED 12 will blink shortly.

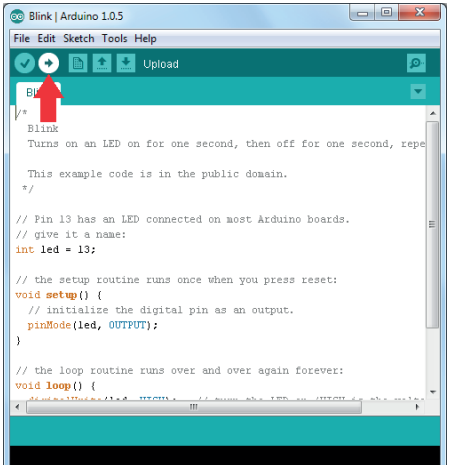


Fig. 6a) Uploading software

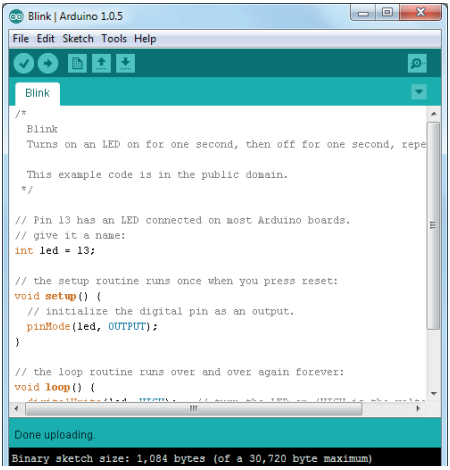


Fig. 6b) Ready uploading

At this stage you can 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.arexxx.com
www.roboternetz.de

4.9 TROUBLE SHOOTING

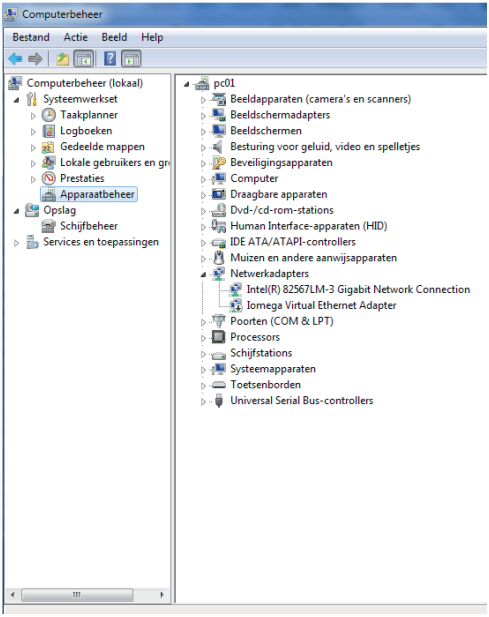
FIRST check if the robot gets enough power from the USB port. You can simply check this by removing the batteries and close jumper JP4, the Power LED should be on now.

PROBLEM; no connection with the AAR robot.

Please check if the data LEDS on the robot (LED 3 and 4) blink shortly when you connect the USB cable with the robot.

Check if the robot is connected with the PC in your device manager.

Fig. 6c)
Device manager (Dutch WIN7)



The Robot FTDI port is normally shown under ports as **"USB SERIAL PORT"**

Check if the correct FTDI driver is installed.

This is most likely the problem. Download and install the latest FTDI driver to your PC.

PROBLEM; CANNOT UPLOAD AN .INO FILE TO THE ROBOT.

Check if you chose the correct COM port, it will be different from COM1 and COM3.

Check if you chose the correct Arduino board type with the correct processor type (ATmega328).

Check your sketch if you have all necesarry libraries installed, see ***Menu entry "sketch-->Library -->*** (see page 22).

4.10 AAR ARDUINO SELFTEST

Please open the AAR_selftest program from CD and download the .ino file into the AAR robot. 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. 7).

Load the libraries

Menu entry “sketch”--> “Library” --> ***SoftPWM*** and ***TimerOne*** from the CD (examples).

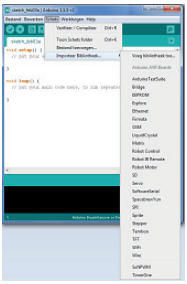


Fig.6d) libraries

Open libraries (dutch version)

Fig. 7 Uploading software

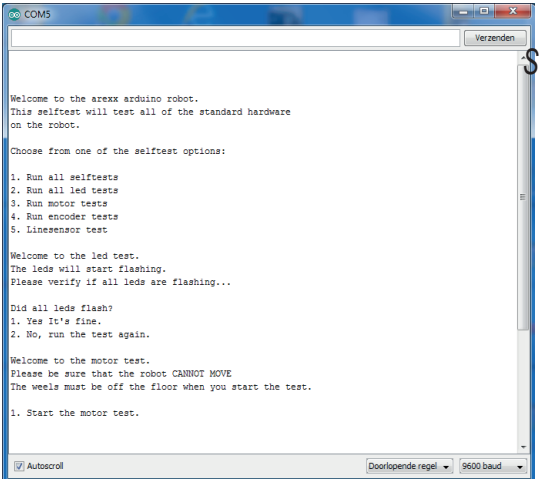


Fig. 8 Serial monitor

Open the serial monitor as shown in Fig 8.

Keep the USB-cable connected to the robot!
Be sure batteries are placed and full.
Switch on the power of the robot.

Start with the first test and follow the steps on the serial monitor.

5 ARDUINO ROBOT EXTENSION KITS

5.1 ASURO Extension kits for the Arduino Robot.

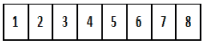
The AREXX Arduino robot has different connectors for extension modules. The AAR connectors are also pin compatible with the AREXX ASURO robot (a robot kit which is programmed in C-language). This means that you can use all the AREXX ASURO kits for this Arduino robot as well.

On the CD we also provide some Arduino example programs, so you can start with the extension kits directly.

The ASURO robot has also a general experiment kit (ARX-EXP2) which can be used for your own Arduino extensions.

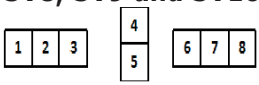
Below please find the PIN layout for the AAR expansion kit connector. These are the connectors SV8, SV9 and SV10 on the board.

SV1



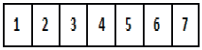
1. D8
2. D4
3. A3
4. GND
5. VCC
6. A2
7. TXD
8. RXD

SV8, SV9 and SV10



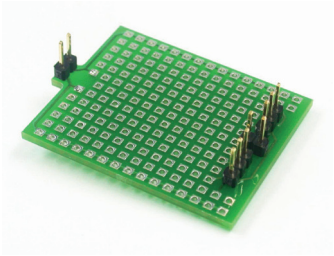
1. SCK/PB5
2. SCL/ ADC5/PC5
3. WS_RIGHT/INT0/PD2
4. RGND
5. PWM1_L/AIN0/OC0A/PD6
6. MOSI/OSC2A/PB3
7. MISO/PB4
8. SDA/ ADC4/PC4

SV2



1. SET
2. AUX
3. TXD
4. RXD
5. EN
6. VCC
7. GND

ARX-EXP2
AREXX EXPERIMENT BOARD



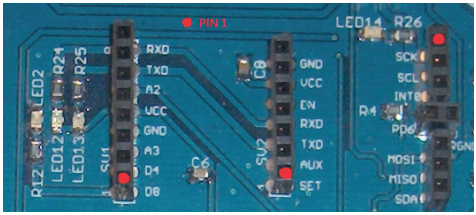


Fig. 9. Extension connectors

Red dot is pin 1

5.2 ASURO Extension kits configurations.

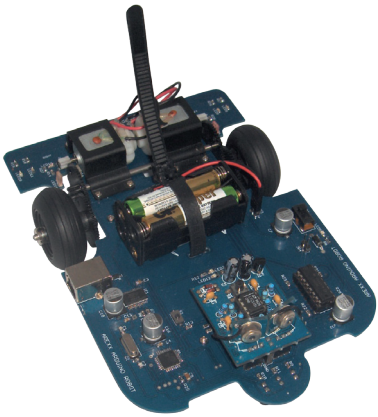
ARX-ULT10
AREXX ULTRA SOUND KIT

Please use the Arduino example program ***AAR_ULTRA*** on the CD. With this sample program the robot drives backward when it detects an object.

ARX-SNK20
AREXX SNAKE VISION (PYRO) KIT

Please use the Arduino example program **AAR_SNAKE** on the CD.

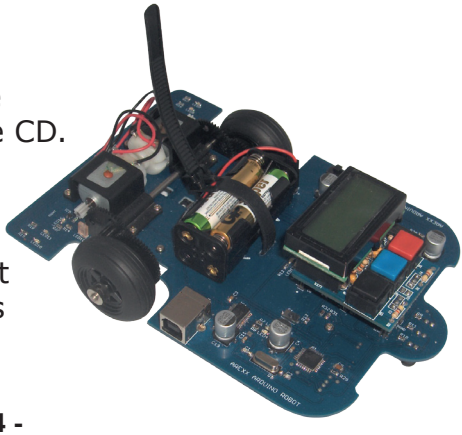
With this sample program the robot will follow a warm object.



ARX-DSP30
AREXX DISPLAY KIT

Please use the Arduino example program **AAR_DISPLAY** on the CD.

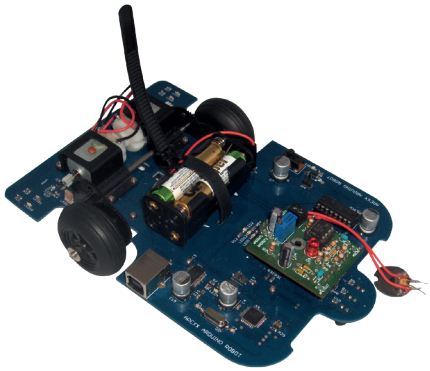
This sample program shows text on the display, the push buttons are freely programmable.



ARX-MNSP55
AREXX MINE SWEEPER KIT

Please use the Arduino example program **AAR_MINE** on the CD.

With this sample program the robot can detect metal.

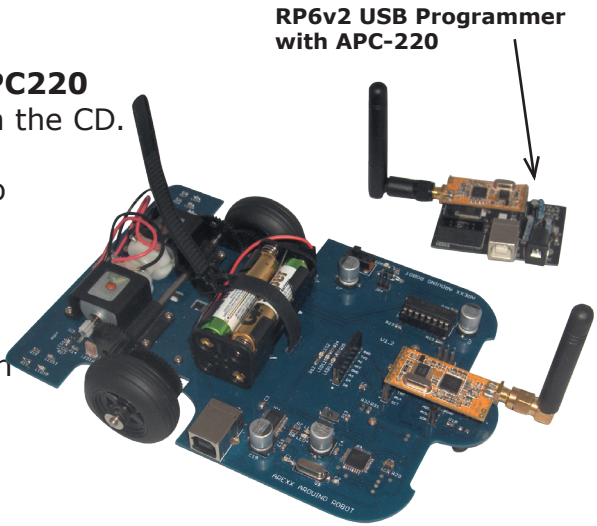


ARX-APC220
AREXX WIRELESS KIT

Software see **AAR_APC220** in the example files on the CD.

Please use the Arduino example program **AAR_SRC.**

You also need the Visual Basic application program for the PC **(AAR_VB)**.



With the AREXX ARX-APC-220 set, you can control the AAR wirelessly.

Below we describe step by step how it works:
- Connect the RP6v2 USB Programmer and the APC-220 to the PC.
- Connect the other APC to the AAR (middle connector).
- Upload the **AAR_SRC** data into the AAR processor.
- Install the **AAR_VB** software on your PC.
- Select the correct COM port in the AAR-VB software.

ANDROID – PROGRAM

ARX-BT3
AREXX BLUETOOTH KIT

Our ARX-BT03 set enables you to control the AAR Robotby Bluetooth with an ANDROID application.

Please open the file **AAR_Android app** on the CD and install the **BLUETOOTH_AAR.APK** on your device and install from the same file the Bluetooth source file AAR_SRC_Bluetooth on the AAR robot.

Follow the step by step instructions below:

- Connect the Bluetooth Module to the PCB of the robot.
- Upload the ANDROID Bluetooth data into the robot processor.
- Upload the ANDROID APK data into your smartphone or tablet.

Bluetooth®

You can find the AREXX android apps on CD **AAR_Android app** or on GOOGLE PLAY.

ANDROID

6. Background information on 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.

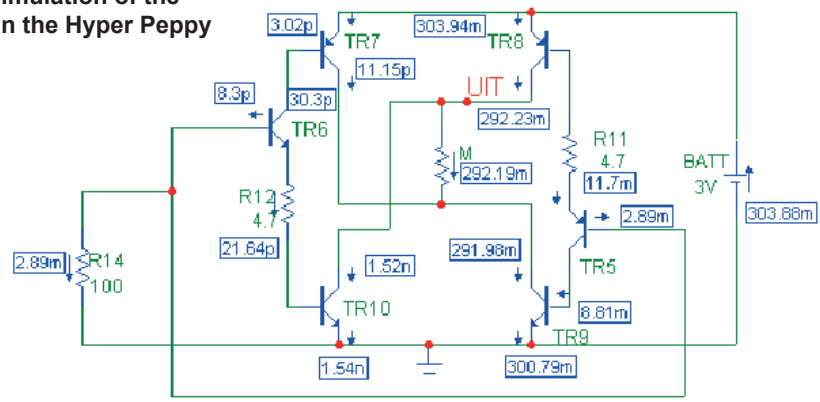
6.1 An H-Bridge for 3 Volt Power supplies

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:

Fig. 10: Simulation of the H-bridge in the Hyper Peppy robot

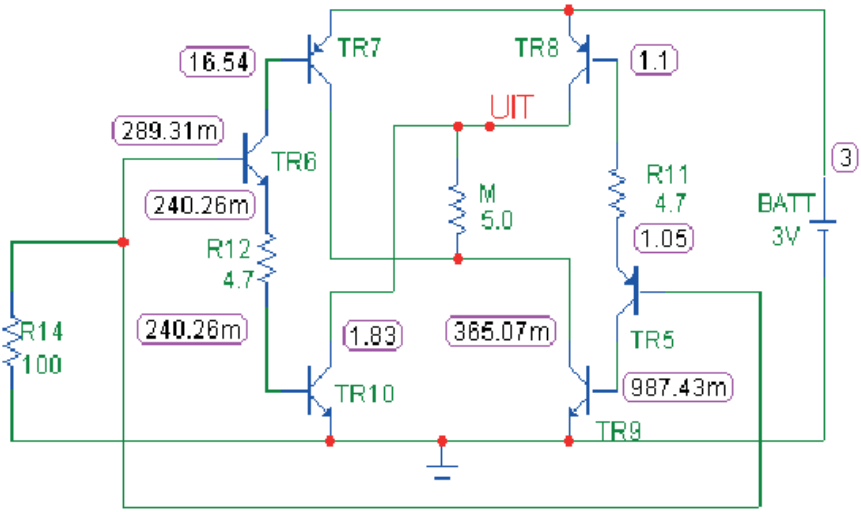


In the driver stage we may identify the DC-motor M. The preamplifier of the driver circuit is being simulated by resistor R14. This resistor will pull the base-ports of transistors TR6 and TR5 to 0V, which results in a condition in which only the right-sided branch is conducting a significant current.

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 approx. 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. 11: 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.

6.2 An H-Bridge for 4,5 Volt

The L293D-chip (see fig. 10) allows us to control output-currents up to 600mA pro 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.

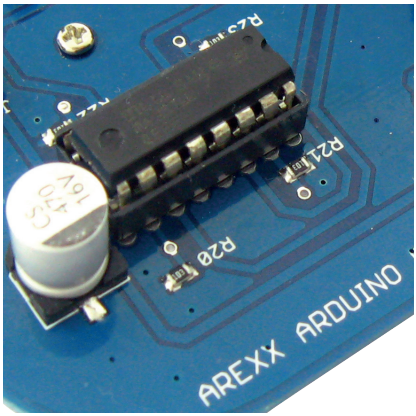


Fig. 12 H-Bridge circuitry with a L293D-Chip

7. 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.

7.1 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 is 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 are considered as practical survival patterns for simple living organisms such as worms.

7.2 Cowards and music-lovers

Quite interesting also is 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 be attracted by the source of the high flute-music in spite of the heavy-metal 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.

7.3 Complex Line-seekers

Line-seeking and line-following robots usually will need a light source such as a LED and twi 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.

7.3.1 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....

8. 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 (for the STK500)
* 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. 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.

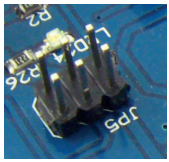


Fig. 12: ISP connector

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

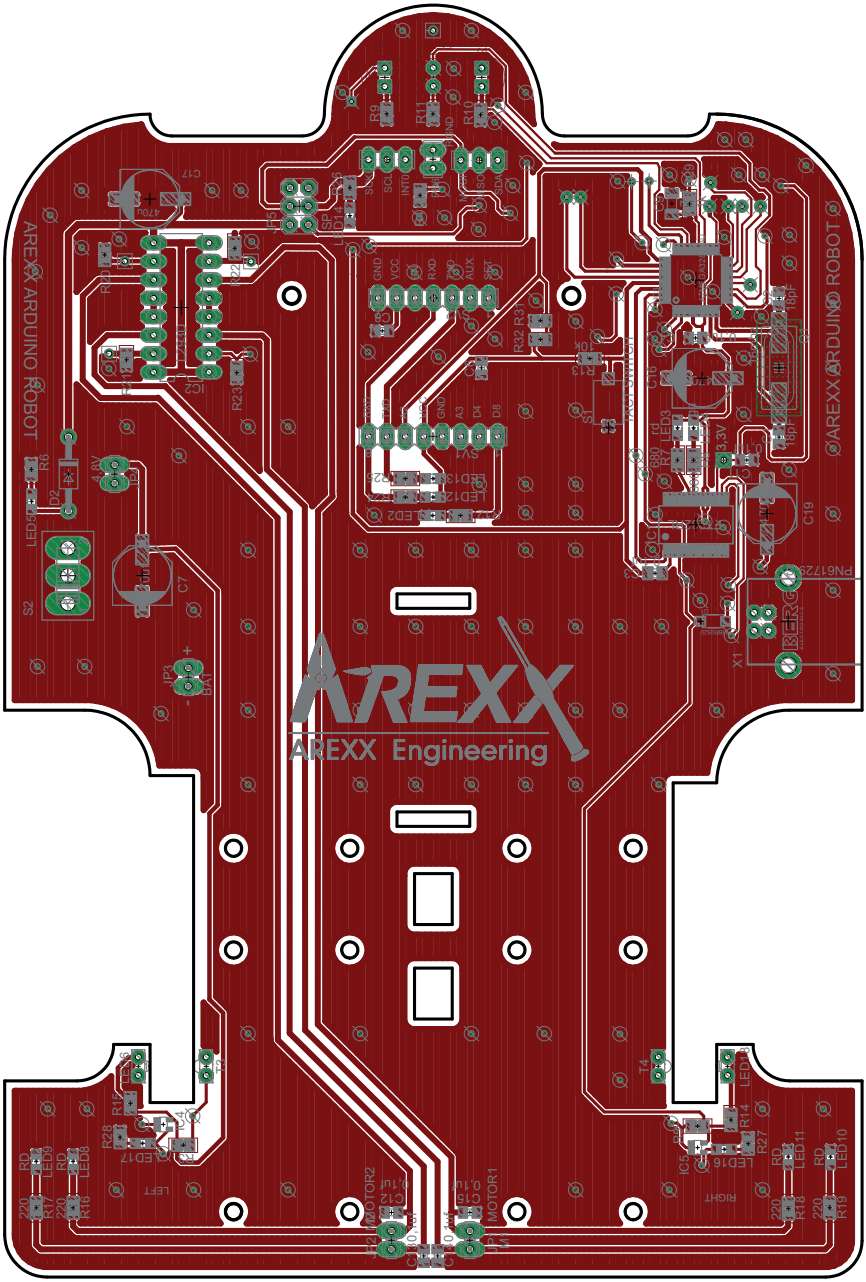
APENDIX

A. Partlist

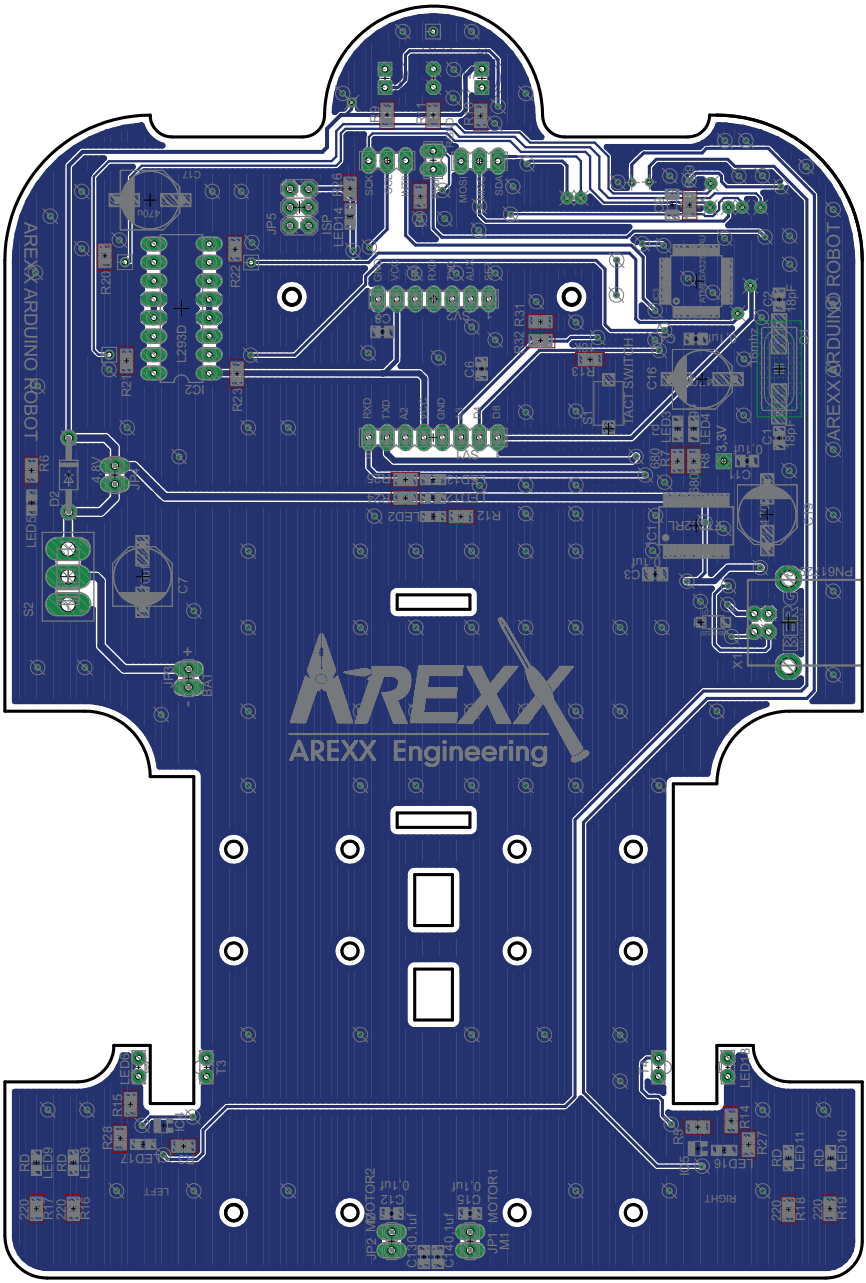
Part	Value	Package
C1	18pF	0805
C2	18pF	0805
C3	0.1uF	C0805K
C4	0,1uF	0805
C6	0,1uF	0805
C7	470uF	CPOL-USF
C8	0,1uF	0805
C9	4,7uF	1206
C11	0,1uF	0805
C12	0,1uF	0805
C13	0,1uF	0805
C14	0,1uF	0805
C15	0,1uF	0805
C16	470uF	CPOL-USF
C17	470uF	CPOL-USF
C19	470uF	CPOL-USF
D1	MBR0520	SOD-123
D2	1N4001	DO41-10
IC1	FT232RL	SSOP28
IC2	L293D	DIL16
IC3	ATMEGA328AU	ATMEGA328P-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

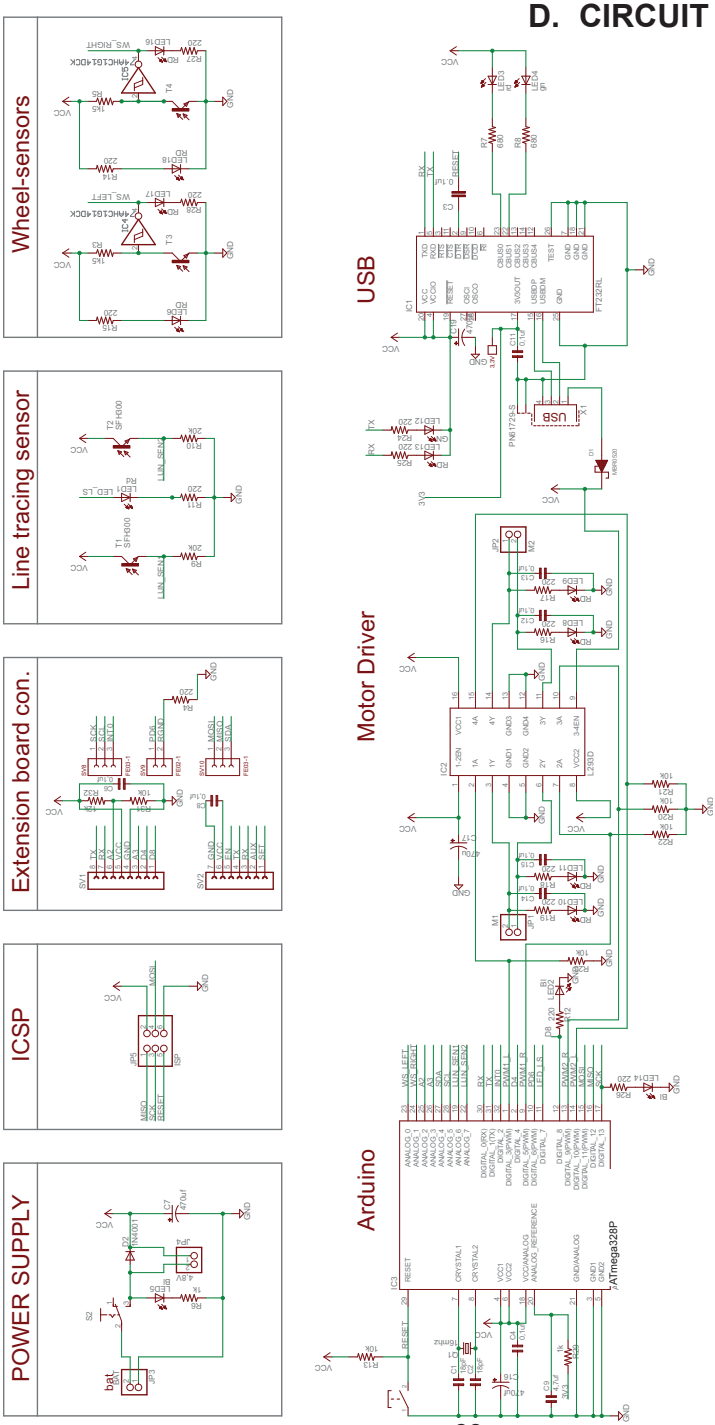
B. MAIN PCB TOP



C. MAIN PCB BOTTOM



D. CIRCUIT AAR



E. 3D PCB AAR