

Walking robot **YETI**

Manual: Model YT-3000



CONTENT

1.	Produkt description YETI	3
2.	YETI general information	4
3.	Who or what is YETI?	11
4.	Hardware	13
6.	Assembly instructions electronics	17
7.	Partlist mechanics	26
8.	Mechanical assembly instructions	29
9.	Charching YETI batteries	41
10.	Software installation and initial steps	48
11.	Preparation for operation	63
12.	YETI Calibration	66
13.	YETI Programming	69
14.	Extensions and upgrade modules	85
xx.	APPENDIX	104
A.	Overview of Yeti functions	105
B.	Diagram YETI	109
C.	Diagram Display module	110
D.	Diagram US module	111
E.	Diagram RS232 IR	112
F-	Diagram USB IR	113
G.	Flatcable connections	114
H.	Error tracking	115
I.	Installing upgrade kits	117
J.	Calibration- and test program	118
K.	Akku ADC values	121

NOTICE!

YETI ROBOT KIT 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 notice.

We are not liable for disadvantage or damage caused by improper use or assembly.



Manufacturer:
AREXX Engineering
JAMA Oriental



European Importer:
AREXX Engineering
ZWOLLE, HOLLAND

Technical support:

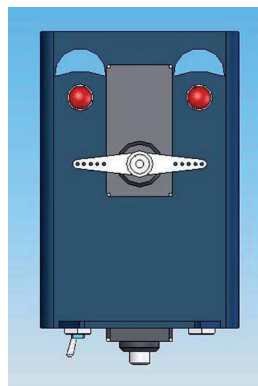
WWW.AREXX.COM
WWW.ROBOTERNETZ.DE

1. PRODUCT DESCRIPTION YETI

1.1. What kind of robot is the YETI?

YETI is a walking robot man, named after “the abominable snowman”, a hairy giant, who is supposed to live in the Himalayan Mountains. Just like the legendary giant, our robot is walking on two big feet.

Our YETI however is supplied with a microprocessor brain and a computer program, controlling a number of servomotors to move its legs and feet.



1.2. Specifications:

YETI ROBOT

Motor:	2 servo motors (5 Volt)
Processor type:	ATmega8L
Programming language:	C
Power source:	4 pcs. AA Accu 4.8 - 6 Volt
Power consumption:	10 mA minimum 600 mA maximum
Communication:	Infrared and I2C bus
Extensions:	Possibility for two extensions by a flatcable
Height	278 mm
Width	155 mm
Depth	100 mm

2. YETI GENERAL INFORMATION

2.1. Who or what is YETI?

As already mentioned YETI is a walking robot, named after “the abo-minable snowman”, a hairy giant, who is supposed to live in the Himalayan Mountains.

Just like the legendary giant, our robot is walking high on his two big feet. Our YETI however is supplied with a microprocessor brain and a computer program, controlling a number of servomotors to move its legs and feet. So it can walk forwards and backwards and even make left and right turns.

For each step YETI must move one feet followed by he other just like humans do. Therefore he needs two servomotors controlled by a brain (microprocessor). A servomotor has an internal gearbox and because of this it is a very powerful motor.

To control the motor rotation and speed, the servomotor also contains an electronic circuit which is a Puls Width Modulation (PWM) control. This PWM control allows the servomotor to make a very accurate rotation step.

YETI has a front servo and a bottom servo. The servo at the front pul moves the feet upwards. The forward and backward move is made by the bottom servo. In this manual another name for bottom servo may also be leg servo!

2.2. How can we use the YETI robot?

- Load different example programs into the YETI.
- Load self-made programs into the YETI.
- Add assembled electronic extension units to enable the YETI to avoid obstacles or to measure distances.
- Add self-made electronic extension units.
- Let YETI communicate with your computer by wireless infrared signals.
- Control YETI by wireless infrared signals from your computer or from your remote TV-control set.
- Let YETI play melodies or make sounds
- Switch off and on his LED-eyes.
- Modify the YETI hardware and YETI's face, for example by a display or a LED mouth.

2.3. YETI may be activated by 3 simple steps

1. Assemble the mechanic and electronic modules of YETI following the instructions in the manual.
2. If you use accumulator-packs, charge the rechargeable devices to 100 %.
3. Activate the main YETI switch at the bottom of the robot.

A few seconds later YETI will stretch his legs and proceed by giving a demonstration of his abilities, following a standard example program in the processor's memory.

Now this does not seem to be so difficult and it looks like we are ready by now. But the real work is still to come and this is just an overture.

We may now concentrate on writing our own programs and modify the robot's abilities and looks in a more creative phase.

2.4. Loading an example program into the YETI memory

We will be using harmless invisible infrared light signals to load an example program into the YETI memory. In fact, the supplied COM-port adapter contains a RS232-infrared transceiver, which must be connected to the COM-port of your PC. The YETI robot contains a built-in infrared transceiver, being located behind both small openings at the robot's back. The computer COM-port adapter is available as an USB-module as well. The ASURO robot, another programmable robot in our robotic range, is using the same IR-transceiver system.

Loading a program into the YETI memory will overwrite the existing program in YETI's memory. The default example program in the memory will be deleted. This however is no problem, as it may be reloaded from the PC into the memory at any time.

2.5. Loading a YETI program into the robot

- Connect the COM-port adapter (or the USB adapter) to your PC.
- Start the Flash program at the computer.
- Select the COM-port for the COM-port adapter in the Flash program at the computer.
- Select the YETI program in the Flash program.
- Allow the holes in the YETI's back to be directed to the upper side of the COM-port adapter.
- Switch off YETI's main switch.
- Press the button "Programming" in the Flash program.
- Switch on YETI's main switch (within 10 seconds).
- Now the YETI program will be transferred into the YETI processor memory.
- Wait until the YETI program has been loaded.
- Switch the YETI off and then on again.
- Wait for 3 seconds.

If everything is OK, YETI will now start executing its new program.

2.6. Add-on kits

You may easily add extra (but non included) modules to the YETI robot, to allow YETI to do much more. How about adding an ultrasonic transceiver, allowing YETI to sense the distance to obstacles and avoid these objects while walking around?

You also may add a display to YETI's body to show output messages or data. Add-on kits usually contain a printed circuit board (PCB) and may be supplied with or without electronic components. The add-on kits are made to fit into the robot's head and will be attached to the skull just above his eyes.

Of course you may also design your own PCB's by using the experimental PCB-set. This module will be attached to YETI's head and will be located at the tiny roof.

A flat cable is used to connect the add-on modules to the main PCB (and to the microprocessor's I2C bus in the robot's skull as well), using an I2C-bus system.

2.7. The communication between YETI and PC

Pressing the “Programming”-Button in the Flash Program will activate the Flash program to contact the YETI robot for 10 seconds. If YETI responds, the YETI program will be transferred. YETI however will only respond to the PC if the contact is tried within a 3 seconds period following YETI’s switch on. If YETI is not being contacted within the 3 seconds period, the robot will proceed by starting the program in its memory. In case the YETI does not respond, the Flash program will output an error message after 10 seconds.

Reports from ASURO-users describe potential problems in data transfers. Especially the RS-232 adapter may be causing difficulties by generating error messages.

These problems may be avoided by:

- Providing a good visible contact between IR-transmitter and -receiver.
- Using the latest Flash software release.
- Using fully charged battery packs.
- Screening the systems from artificial light sources (especially fluorescent lamps).
- Using an USB-adapter (especially in case the RS-232 voltage is low).

4. WHO OR WHAT IS YETI?

4.1. Summary

As discussed in the first chapter the YETI is a mythical figure, the abominable snowman, living in the Himalya Mountains and clumsily walking on two big feet.

Our YETI is a tall, upright standing robot, equally walking on two big feet and being able to move forward or backward or make a left or a right turn.

Every step forward or backward will start by balancing on one foot and moving the opposite foot. Basically two servos will be executing these movements. A servo is a special motor type, including a gear. Gears are used to reduce speed, but will equally increase the motor's torque. Additionally the servo is equipped with impulse-controlled electronics, allowing an exact positioning of the servo's rotation angle.

The YETI is equipped with two servomotors, located at the frontside and at the bottom side. The servo at the frontside rises YETI's feet in order to move them (and is named the "feet-servo") and the servo at the bottomside moves the legs (including the feet) one by one (the "legs-servo").

4.2. The basics of walking

YETI is balancing on one foot and is moving the opposite foot. To do so YETI pulls the outer side of the "balancing" foot upwards and simultaneously presses the outer side of its "moving" foot downwards. These movements will bend YETI's body over to the "balancing" foot, which is now carrying the main part of the robot's weight. In a next phase YETI will shift its "moving" foot forward, completing the movement process. The "moving" foot is now turning into a "balancing" foot and vice versa.

In the walking process these movement are constantly being repeated.

Detailed explanation

Starting from stillstand, the YETI starts rotating the feet servo clockwise as seen from the front side. These rotations will result in two movements:

In a first phase the right side of the feet servo is rising, pulling the outer side of the right foot upwards. This action might bend the body to the right, but this movement is impossible as long as the main part of the weight is still resting on the inner side of the right leg.

At the same time the left side of the feet servo is lowering, pressing the outer side of the left foot downwards. This action is lifting the left leg upwards, bending the body to the right and transferring the main part of the weight to the right foot.

If YETI raises his right foot a little bit too much, the left foot will bend YETI's body over the equilibrium point, causing the robot to fall down to the right side. YETI must raise his right foot just high enough to prevent the left leg from pushing him beyond the equilibrium. The closer the body reaches the equilibrium, the greater percentage of the weight is resting on the right foot and the easier the robot is able to move his left foot.

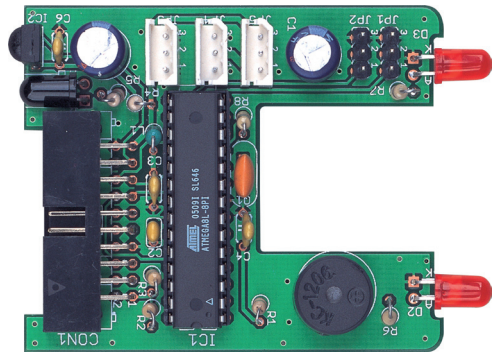
At this point the legs servo is activated. The legs servo moves YETI's left leg forward as far as possible. While moving the left foot forward from behind, the body will also be bended up to a position halfway the left foot, but still following the line of the right foot. YETI is still balancing on his right foot. As soon as the left foot has reached its ultimate position, the robot raises the outer side of the left foot and simultaneously presses the outer side of the right foot downwards. These actions will move the body and its weight as well (including the equilibrium point) from the right foot to the left one. Again we must be careful to prevent YETI to move too far to the left, which would result in falling down.

The movement phases will symmetrically be repeated for each single leg.

4. HARDWARE

4.1. YETI main PCB

YETI's main printed circuit board (PCB) is equipped with an Atmega8L microcontroller chip. The microcontroller is connected to a big, round beeper and to both red LED's at the front side. A great number of other microcontroller pin connections, e.g. the I2C bus, are routed to the 20-pin connector at the rear side directly. This connector provides an extension port to other hardware by a flat cable. Both black connectors next to the red LED provide the connections to both servos. Three white connectors serve the connectors for the main switch and for both battery packs.



4.2. Flat cable connections

The designers reserved four of the twenty flat cable connections. Pin 19 is used for VCC and pin 7, 8 and 20 for GND.

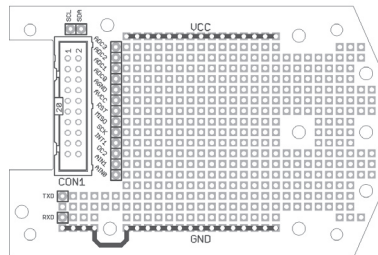
All remaining flat cable connections are connected to a dedicated microcontroller pin. The pin numbers are listed with the assigned microcontroller pin functions.

Microcontroller pins may be used in several modes, depending on the programming mode. Please check the microcontroller's datasheet or manual for details.

4.3. Flat cable connections

More info about flat cable connections: (see page. 20)

Pin 1	SCL
Pin 2	SDA
Pin 3	PC3(ADC3)
Pin 4	PC2(ADC2)
Pin 5	PC1(ADC1)
Pin 6	PC0(ADC0)
Pin 7	GND
Pin 8	GND
Pin 9	AVCC
Pin 10	PC6(RESET)
Pin 11	PB5(SCK)
Pin 12	PB4(MISO)
Pin 13	PB3(MOSI/OC2)
Pin 14	PD3(INT1)
Pin 15	PD6(AIN0)
Pin 16	D7(AIN1)
Pin 17	PD0(RXD)
Pin 18	PD1(TXD)
Pin 19	VCC
Pin 20	GND



4.4. YETI experimental extension set

Summary

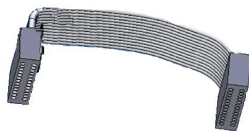
The experimental extension set has been designed to set up your own electronic designs and connect the experimental modules to the micro-controller.

4.5. Flat cable connections

The designers reserved four of twenty flat cable connections. Pin 19 is used for VCC and pin 7, 8 and 20 for GND.

All remaining flat cable connections are connected to a dedicated microcontroller pin. The pin numbers are listed with the assigned microcontroller pin functions. Microcontroller pins may be used in several modes, depending on the programming mode. Please check the microcontroller's datasheet or manual for details.

Pin 1	SCL	Serial Clock (for I2C communication)
Pin 2	SDA	Serial Data (voor I2C communication)
Pin 3	PC3(ADC3)	Digital input/output or analog monitor input
Pin 4	PC2(ADC2)	Digital input/output or analog monitor input
Pin 5	PC1(ADC1)	Digital input/output or analog monitor input
Pin 6	PC0(ADC0)	Digital input/output or analog monitor input
Pin 7	GND	GND (several connectors to prevent signal noise)
Pin 8	GND	GND (several connectors to prevent signal noise)
Pin 9	AVCC	Analog reference-voltage for AD-converters
Pin 10	PC6(RESET)	Microcontroller reset pin
Pin 11	PB5(SCK)	Digital input/output
Pin 12	PB4(MISO)	Digital input/output or I2C function pin
Pin 13	PB3(MOSI/OC2)	Digital input/output or I2C function pin or Timer2 pin
Pin 14	PD3(INT1)	Digital input/output or external interrupt
Pin 15	PD6(AIN0)	Digital input/output or analog testinput
Pin 16	D7(AIN1)	Digital input/output or analog testinput
Pin 17	PD0(RXD)	Digital input/output or RS232 input
Pin 18	PD1(TXD)	Digital input/output or RS232 input
Pin 19	VCC	VCC
Pin 20	GND	GND (several connectors to prevent signal noise)



Flatcable connections on the YETI upgrade kits

Made by uwegw, No guarantee for failures!

YETI Pins soldering side		Cable										US			
Display/I2C		1	3	5	7	9	11	13	15	17	19				
Upgrade kit Number	Signal	PC5/SCL	PC3/ADC3	PC1/ADC1	AGND	AVCC	PB5/SGK	PB3/OC2	PD6/AIN0	PD1/TXD	VCC				
Upgrade kit Number	Signal	PC4SDA	PC2/ADC2	PC0/ADC0	AGND	RESET	PB4/MISO	PD3/INT1	PD7/AIN1	PD0/RXD	GND				
Display/I2C		2	4	6	8	10	12	14	16	18	20				
		US													
		PCB													

Upgrade kit
Number
Signal
Signal
Number
Upgrade kit

YETI Pins Component side		Cable										US			
Display/I2C		2	4	6	8	10	12	14	16	18	20				
Upgrade kit Number	Signal	PC4SDA	PC2/ADC2	PC0/ADC0	AGND	RESET	PB4	PD3/INT0	PD7/AIN1	PD0/RXD	GND				
Upgrade kit Number	Signal	PC5/SCL	PC3/ADC3	PC1/ADC1	AGND	AVCC	PB5	PB3/OC2	PD6/AIN0	PD1/TXD	VCC				
Display/I2C		1	3	5	7	9	11	13	15	17	19				
		US													
		PCB													

Upgrade kit
Number
Signal
Signal
Number
Upgrade kit

Legend:

VCC	free
GND	used
Upgrades, otherwise free	

Pins on the upgrade PCB

Pin		1	2	3	4	5	6	7	8	9	10	11	12	13
Flatcable No.		15	16	13	14	11	12	10	9	7/8	6	5	4	3
Signal		PD6/AIN0	PD7/AIN1	PB3/OC2	PD3/INT1	PB5/SGK	PB4/MISO	RESET	AVCC	AGND	PC0/ADC0	PC1/ADC1	PC2/ADC2	PC3/ADC3
Upgrade kit		US										US		

Con3 I2C

Pin	1	2
Flatcable No.	1	2
Signal	PC5/SCL	PC4SDA
Upgrade kit	Display/I2C	Display/I2C

Con4 UART

Pin	1	2
Flatcable No.	17	18
Signal	PD1/TXD	PD0/RXD
Upgrade kit	IR	IR

Con5 VCC

Pin	1	2
Flatcable No.	20	19
Signal	GND	VCC

5. ASSEMBLY INSTRUCTIONS ELECTRONICS

First of all please check if all parts in the kit are complete.

5.1. Soldering job

The layout on the PCB clearly shows where to mount each component.

When you want to see more detailed information, please study the diagram, pictures and drawings for extra help and information.

When we assemble a PCB we always start with the lowest passive components. Normally these are the resistors followed by the capacitors. After soldering, cut the wire ends of the components directly, so you always keep enough space for the soldering of the other components on the PCB.

Before you start soldering, we always advise to insert the active components (transistor, IC, diode) so you already can align their pins when they do not fit properly. Often the legs of such components need some extra bending to make them fit. At last you solder the IC sockets or the active components.

IMPORTANT

The Elco and IC have a polarity so you should be careful to solder them in the correct position.

WARNING

NEVER start with the soldering of an IC and when possible always use a IC socket to avoid mistakes. When an IC is soldered, it is very difficult to remove it again.

TIP

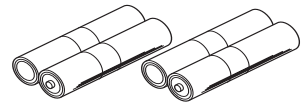
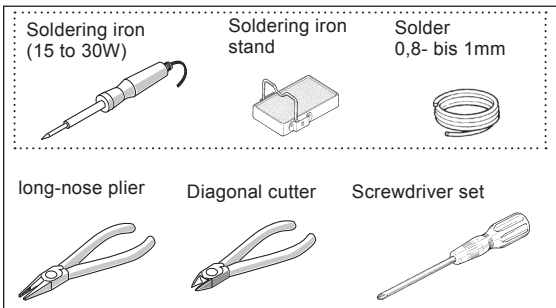
The IC-pins can be bend and aligned very simple on a hard flat surface like a table! Just put all pins inline on the table and bend the IC carefully to the correct alignment.

***Technical questions see; www.arexx.com --> Forum
www.roboternetz.de --> Forum***

CAUTION

- Read this manual carefully in advance to fully understand how to assemble this product.
- Children below 14 should can only assemble this product with the help of adults.
- Be careful about tools. Especially be careful about sharp tools such as nippers or cutter knife to prevent any injuries or accidents.
- Never assemble the kit when a younger child is around. The child might touch sharp tools or swallow parts and a vinyl bag.
- Be careful about sharp edges of parts.
- * Do not mix old and new or rechargeable and non rechargeable batteries.
- Take out the batteries when you do not use the YETI for more than a week
- The specification, shape and size of the product are be subject to change without prior notice.

5.2.Necessary tools



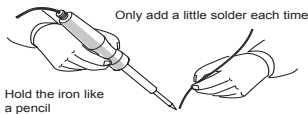
Necessary Batteries:
AA Batteries, 3 Pieces
(not included)

5.3. Soldering techniques:

Only use lead free ROSIN CORE solder!
Never use any liquid- or paste flux!



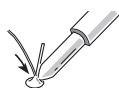
Correct position for professional soldering



1. Preheat the solder area which must be soldered and the component wire with the tip of the iron. Do NOT overheat it!



2. Add some solder to the soldering area and component wire but NOT TOO MUCH!



3. Ziehe den Lötdraht zurück und lasse das Lötzinn richtig fließen.



4. Take away the soldering iron and DO NOT MOVE the component or PCB!



5. Cut away the long component wire just above the soldering spot.



The END RESULT is a nice and shiny soldering spot which is attached to the PCB copper and component wire.



5.4. Troubleshoot soldering mistakes:

Cold PCB



Solder is attached to component wire but not to the PCB copper



Not enough solder



Solder did not flow



Solder bridge



Two separate solder spots are connected to each other



PERFECT SOLDERING



The soldering surface looks nice and shiny



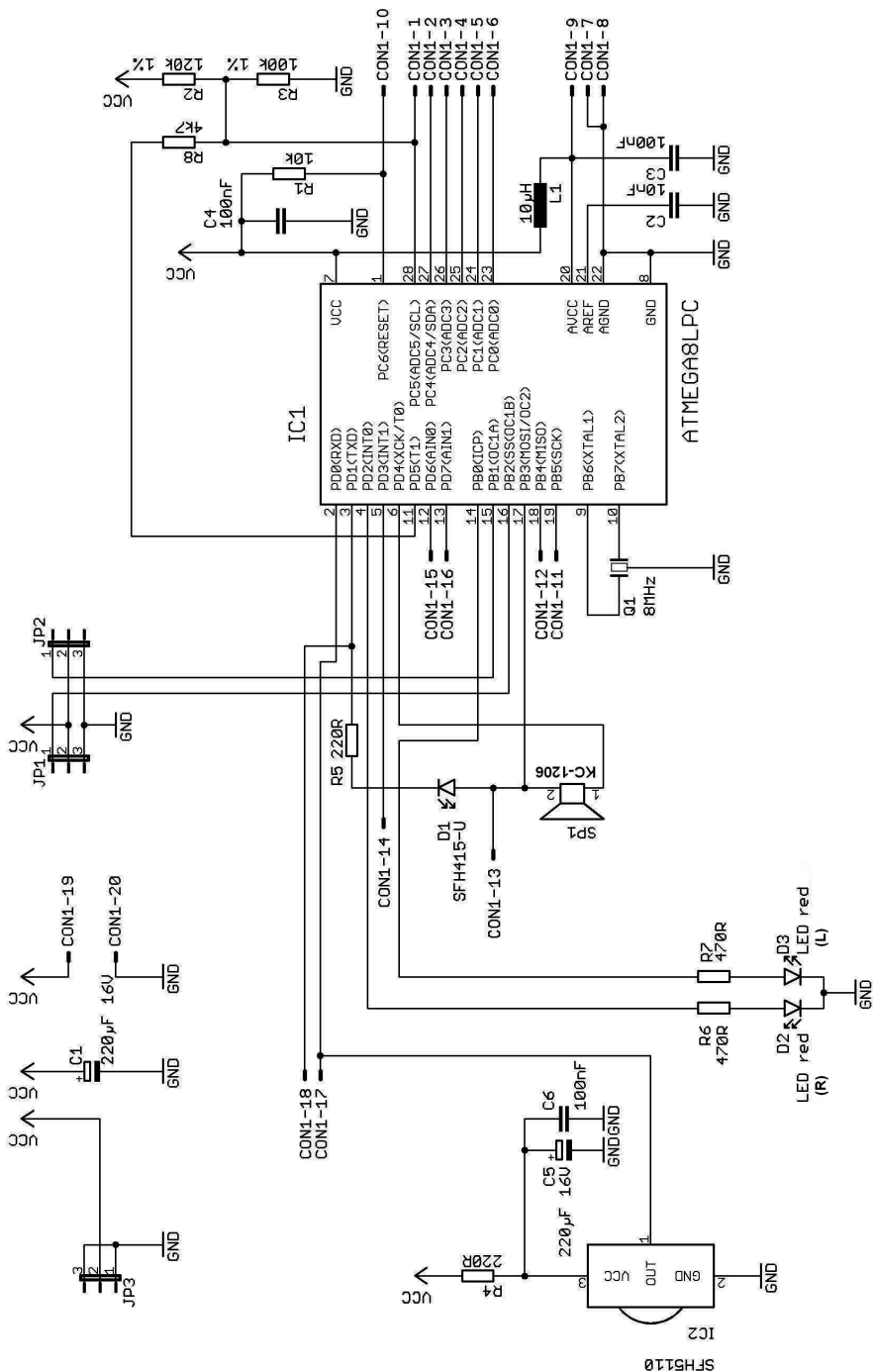
5.4. Assembly main PCB

IMPORTANT see 5.5 (Diagram) and 5.6 (Pictures)

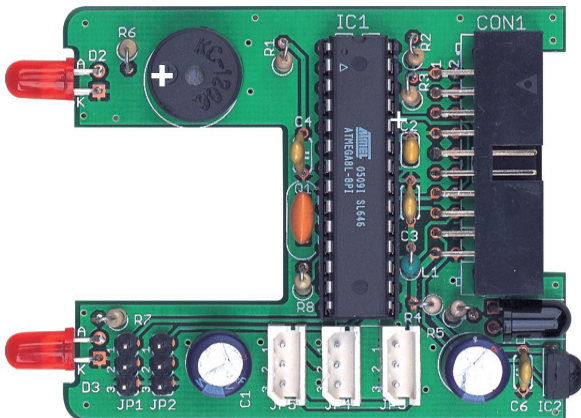
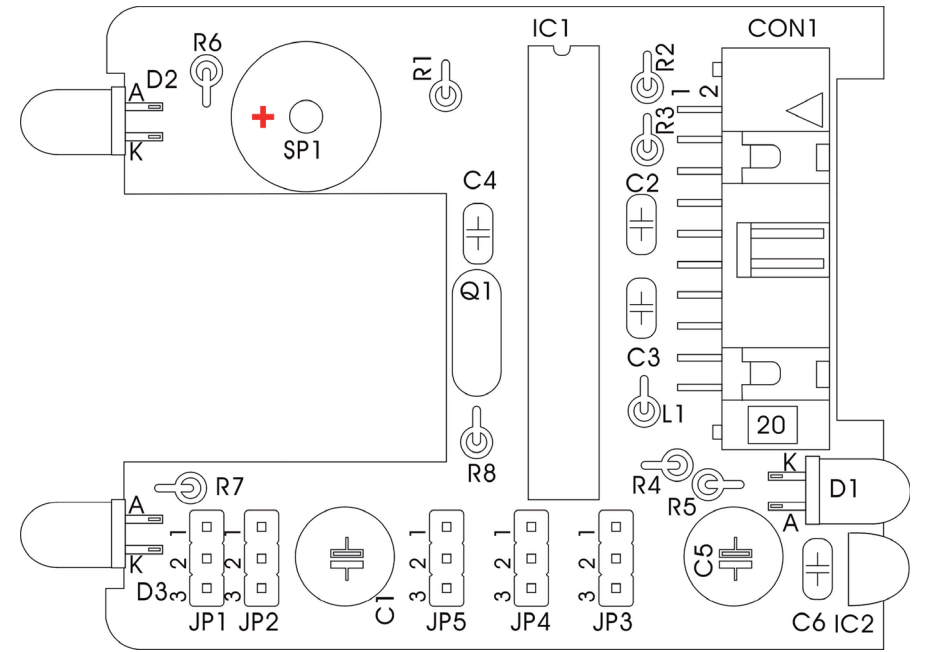
YETI parts list

Nr.	Name	Pcs.
PCB1	Main PCB	1
IC1	ATmega8-I (polarity!)	1
IC2	SFH5110 IR-receiver-IC (polarity!)	1
R1	10K / 0.25W / 5% (Brn, blk, or, gld)	1
R2	120K / 0.25W / 1% (Brn, red, blk, or, brn)	1
R3	100K / 0.25W / 1% (Brn, blk, blk, or, brn)	1
R4	220R / 0.25W / 5% (Red, red, brn, gld)	1
R5	220R / 0.25W / 5% (Red, red, brn, gld)	1
R6	470R / 0.25W / 5% (Ylw, vio, brn, gld)	1
R7	470R / 0.25W / 5% (Ylw, vio, brn, gld)	1
R8	4K7 / 0.25W / 5% (Ylw, vio, red, gld)	1
L1	10uH (Brn, blk, blk, silver)	1
C1	220uF/16V (polarity!)	1
C2	10nF (103)	1
C3	100nF (104)	1
C4	100nF (104)	1
C5	220uF/16V (polarity!)	1
C6	100nF (104)	1
D1	SFH415-U IR-LED (polarity!)	1
D2	LED Red, 5MM (polarity!)	1
D3	LED Red, 5MM (polarity!)	1
Q1	Quarz, 8Mhz / 3 PIN	1
SP1	Beeper, 5V, (KC1206) (polarity!)	1
IC socket	28 PIN, IC socket (polarity!)	1
JP1	3 PIN, PCB type, black	1
JP2	3 PIN, PCB type, black	1
JP3	3 PIN, PCB type, white	1
JP4	3 PIN, PCB type, white	1
JP5	3 PIN, PCB type, white	1
CON1_PCB	Connector, male, 20 pins, for flat cable/ 90 degree angle	1

6.3. Diagram YETI

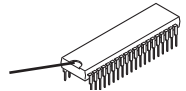


5.6. Hauptplatine



IC

IC Marking



Resistor & Coil

Marking with color stripes



LED & IR-LED

The side with the flat marking is the Cathode.



The long leg is the e Anode.

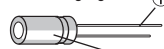
Condensator

No polarity



ELCO (Elektrolyt Condensator)

The long leg is the



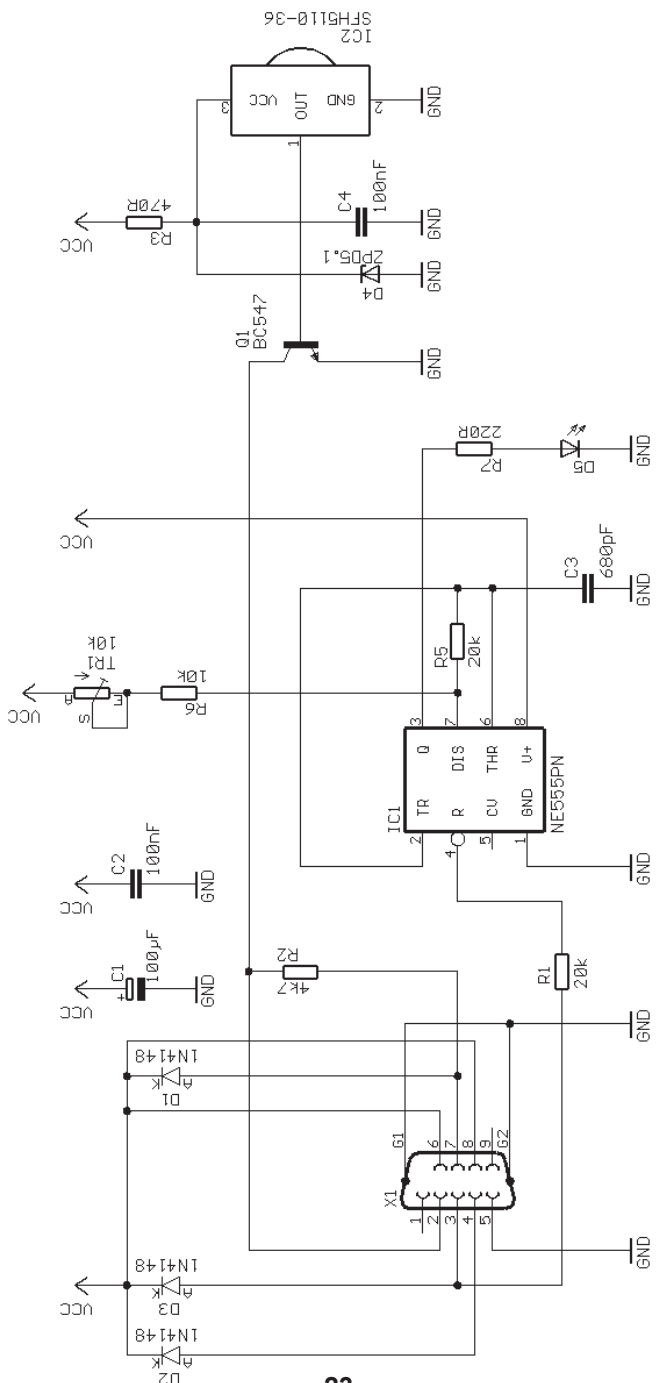
The white line marking on the housing is the



5.7. Assembling the RS232-Infrared-Transceiver

- IC1: Initially insert the 8-pole socket. The polarity mark of the (slightly asymmetrical) socket must correspond with the mark in the accompanying symbol on the PC.
- D1, D2, D3: 1N4148, pay attention to polarity! Read the imprints of the parts and take care not to interchange with ZPD5.1 or BZX55-C5V1!
- D4: ZPD5.1 or BZX55-C5V1, pay attention to polarity! Read the imprints of the parts and take care not to interchange with 1N4148!
- D5: SFH-415-U IR LED (Black LED) pay attention to polarity, press downwards to the PCB
- C1: 100 μ F at least 16 Volt, pay attention to polarity!
- C2, C4: 100nF ceramic capacitor, imprint: 104
- C3: 680pF ceramic capacitor, imprint: 681
- Q1: BC547 (A,B or C) or BC548 (A,B or C)
- R1, R5: 20k Ω (red, black, orange, gold)
- R2: 4.7k Ω (yellow, violet, red, gold)
- R3: 470 Ω (yellow, violet, black, gold)
- R4: Does not exist
- R6: 10k Ω (brown, black, orange, gold)
- R7: 220 Ω (red, red, brown, gold)
- TR1: 10K Ω variable resistor
- IC2: SFH5110-36 Infrared receiver IC, bend the legs with appropriate tongs! Pay attention to polarity (the curvature must be positioned to the outside)!
Caution: electrostatic discharge (ESD) and excessive soldering or heating may damage the part!
- X1: 9pol. SUB-D connector, case must be settled close to PCB. Attachment strips must be soldered as well!
- IC1: insert the NE555P, pay attention to polarity!

5.8. Diagram RS-232 IR-Transceiver



5.9. Layout RS232-Infraread-Transceiver PCB

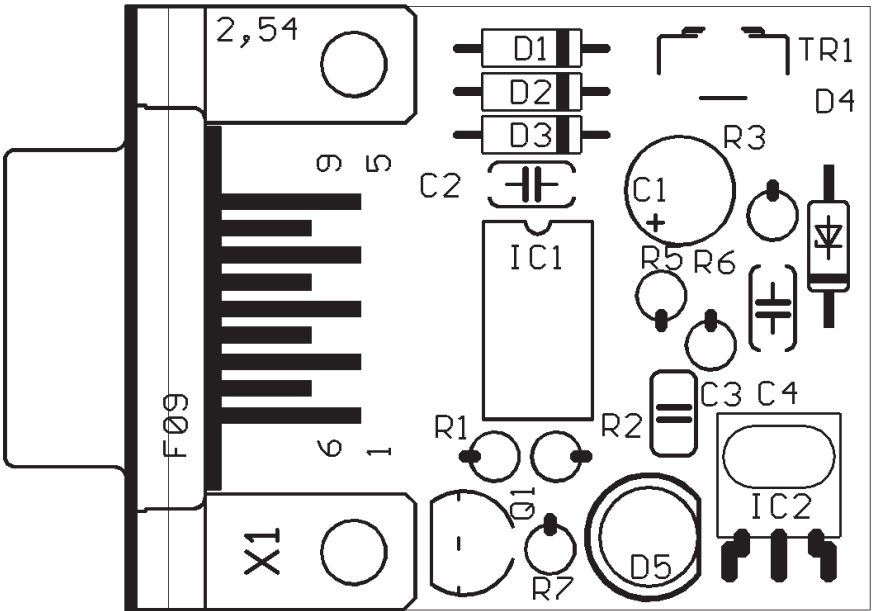
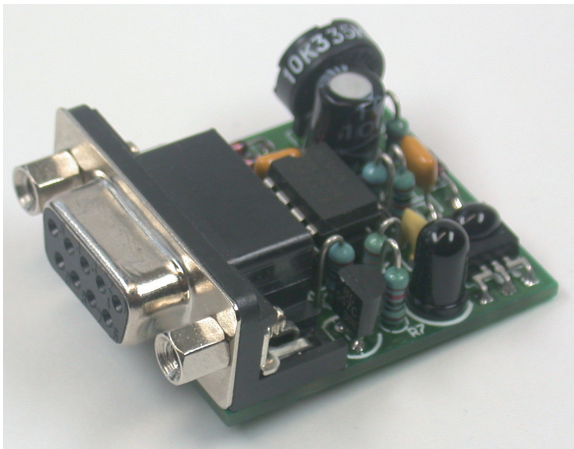


Fig. 5.1.: Layout RS232- Infrared-Transceiver



Finally check the board for short circuits or polarity errors.
Check the soldering quality intensively and re-solder bad contacts.

5.10. Info assembled USB-Infrared-Transceiver

The USB-IR-Transceiver is also available already assembled.

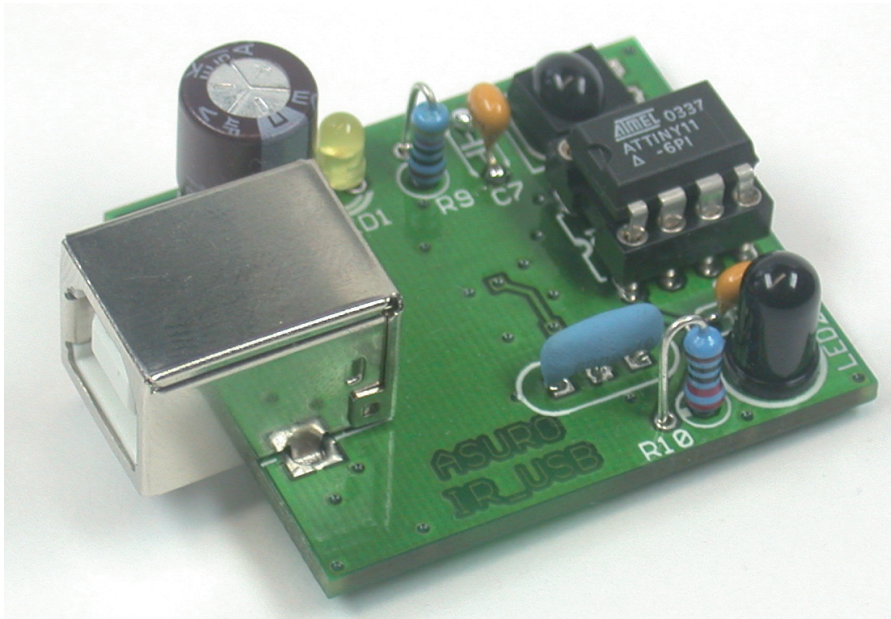


Fig 5.2.: USB Infrared-Transceiver

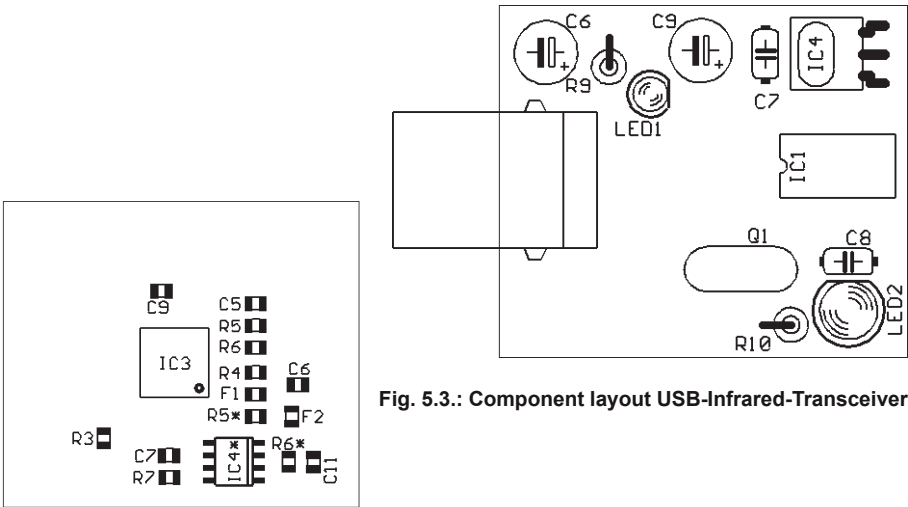

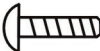





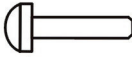




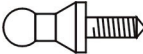
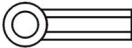


Fig. 5.3.: Component layout USB-Infrared-Transceiver


Fig. 5.4.: Bottom layout USB-Infrared-Transceiver

6. PARTS LIST MECHANICS

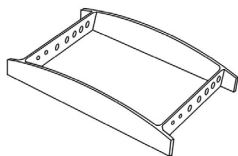
Nut M3	Flathead screw M3x8mm	Collar big	Colar small	Collar screw	Nut M2
					
O 8 pcs.	O 8 pcs.	O 4 pcs.	O 4 pcs.	O 10 pcs.	O 10 pcs.

Spacer	Rivet	Screw-rod connector	Servo Screw
			
O 4 pcs.	O 4 pcs.	O 2 pcs.	O 2 pcs.

Ball-end screw	Ball adjuster	Ballhead screw	HEX-tool
			
O 4 pcs.	O 4 pcs.	O 4 pcs.	O 1 pc.

Thread end rod long	Thread end rod short
	
O 2 pcs.	O 4 pcs.

Foot



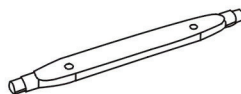
O 2 pcs.

Rod
5 x 80mm



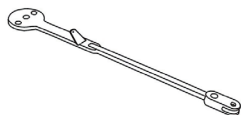
O 2 pcs.

Feet joint panel



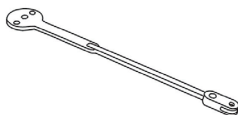
O 2 pcs.

Front leg



O 2 pcs.

Rear leg



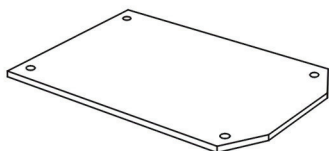
O 2 pcs.

Servo arm pushrod



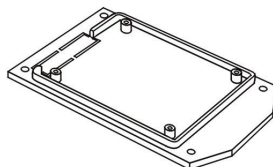
O 2 pcs.

Top cover panel



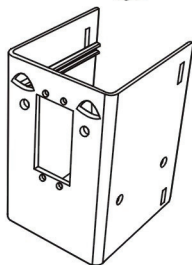
O 1 pc.

Top panel



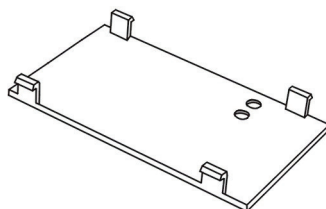
O 1 pc.

Head panel



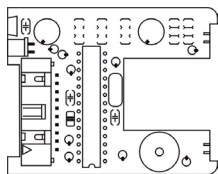
O 1 pc.

Back panel



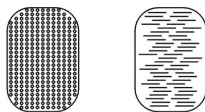
O 1 pc.

**YETI main PCB
(assembled)**



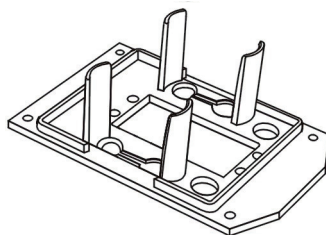
O 1 pc.

Velgro-tape



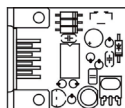
O 2 pcs. Male
O 2 pcs. Female
Pre-assembled

Bottom panel



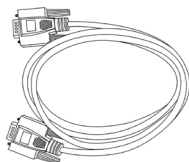
O 1 pc.

**IR/Transceiver PCB
(assembled)**



O 1 pc.

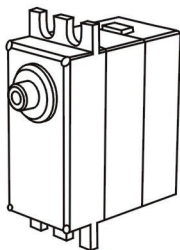
RS-232 Cable



O 1 pc.

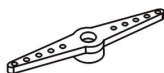
7.1. Important electronic parts

Servo motor



O 2 pcs.

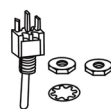
Servo arm



O 1 pc. 1mm
O 1 pc. 2mm

Pre assembled with cableset

Switch



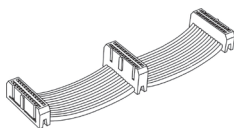
O 1 pc.

DC-connector



O 1 pc.

Flat cable



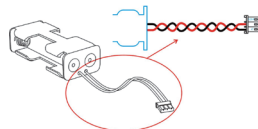
O 1 Pc.

Cable set, Pre-assembled



O 1 pc.

Battery holder



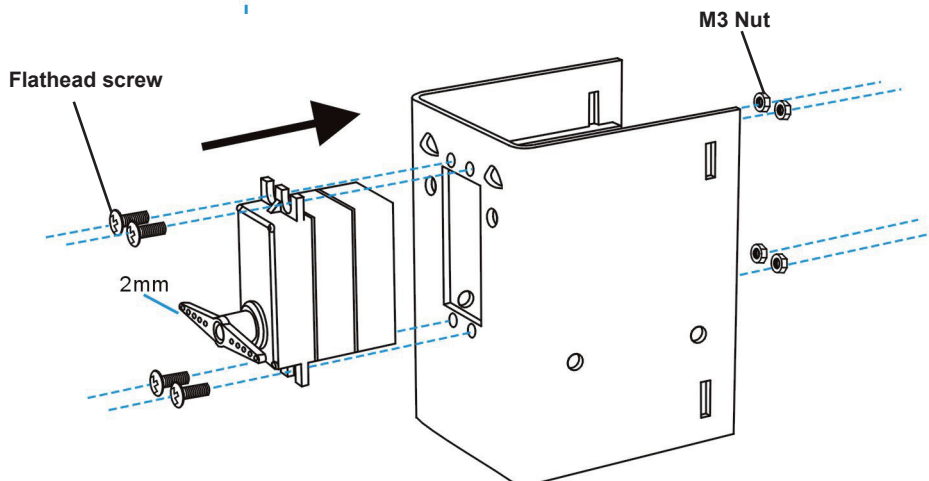
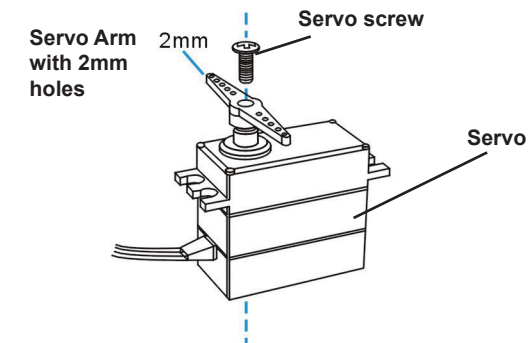
O 2 pcs.

7. INSTRUCTIONS MECHANICAL ASSEMBLY

Installing the head servo:

For this assembly you need;

- 1 pc. Head panel
- 1 pc. Servo
- 4 pcs. Flathead screw
- 4 pcs. Nut M3
- 1 pc. Servo arm 2mm holes
- 1 pc. Servo arm screw



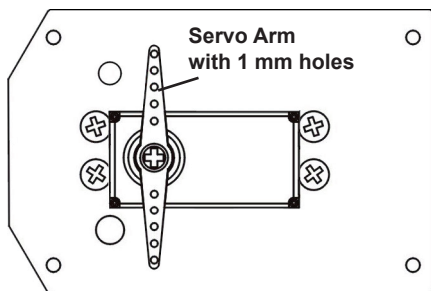
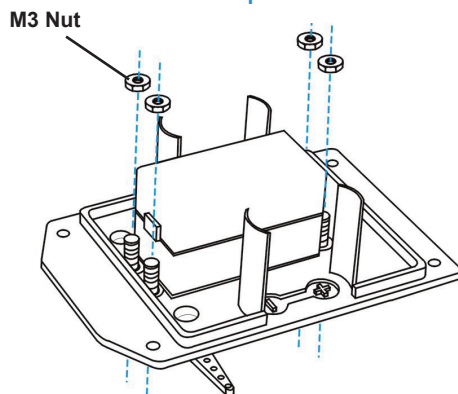
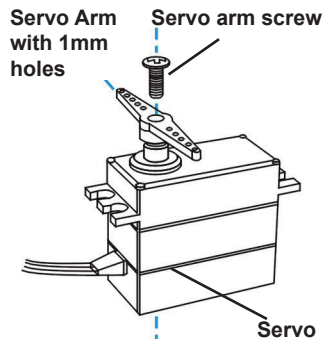
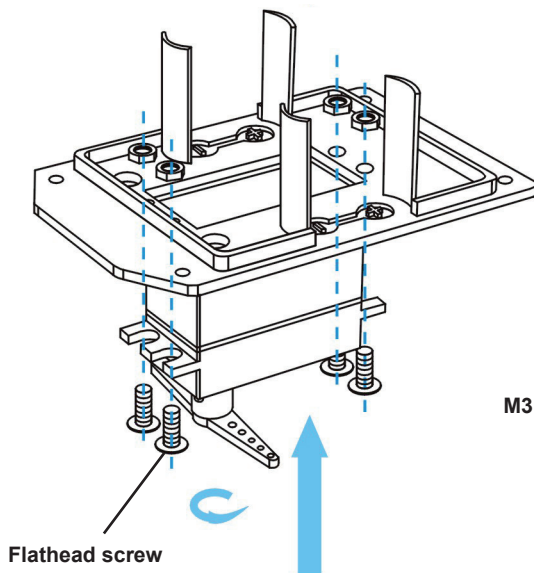
Assemble the servo as shown in the drawings.

Assemble the servo arm to the servo, see the detailed drawing!

Installing the bottom servo:

For this assembly you need;

- 1 pc. Bottom panel
- 1 pc. Servo
- 4 pcs. Flathead screw
- 4 pcs. Nut M3
- 1 pc. Servo arm
- 1 pc. Servo arm screw



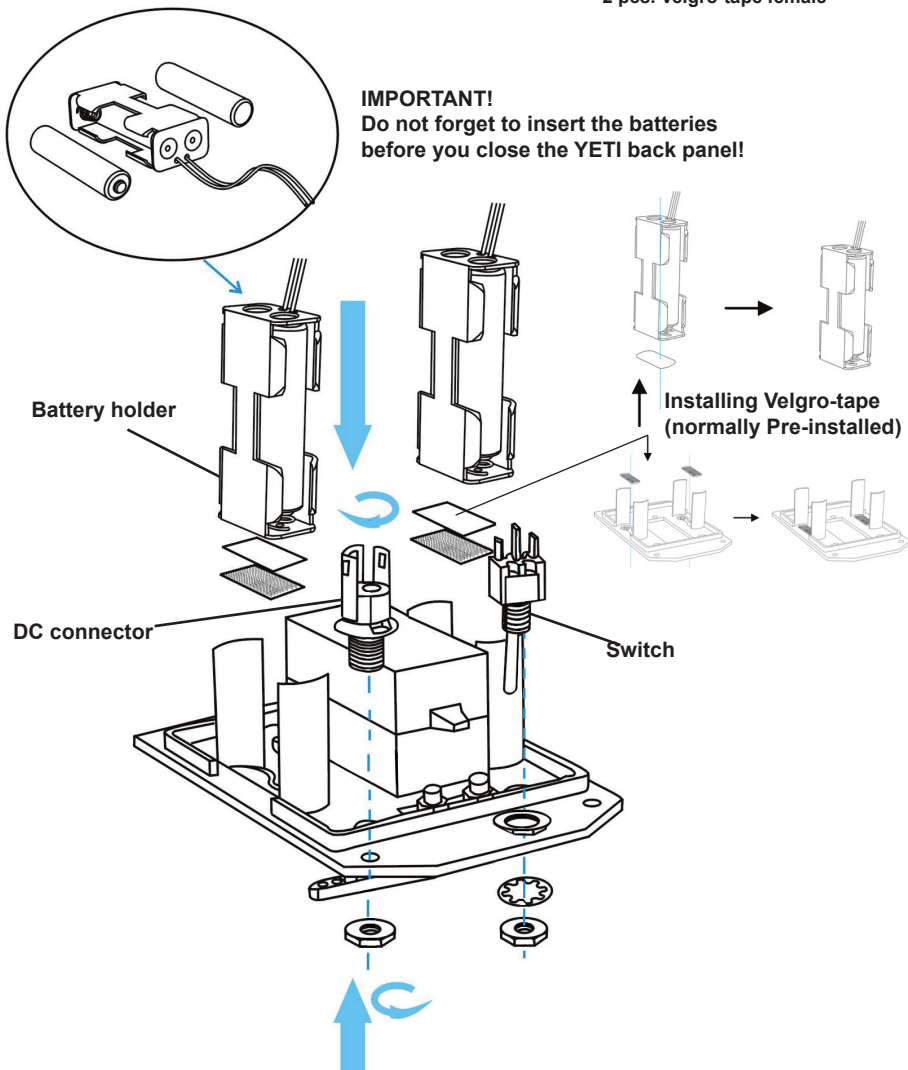
Assemble the servo as shown in the drawings.

Assemble the servo arm to the servo, see the detailed drawing!

End assembly bottom panel:

For this end assembly you need;

- 1 pc. Assembled bottom panel
- 1 pc. Switch
- 1 pc. DC connector
- 2 pcs. Battery holder
- 4 pcs. AA accu
- 2 pcs. Velgro-tape male
- 2 pcs. Velgro-tape female

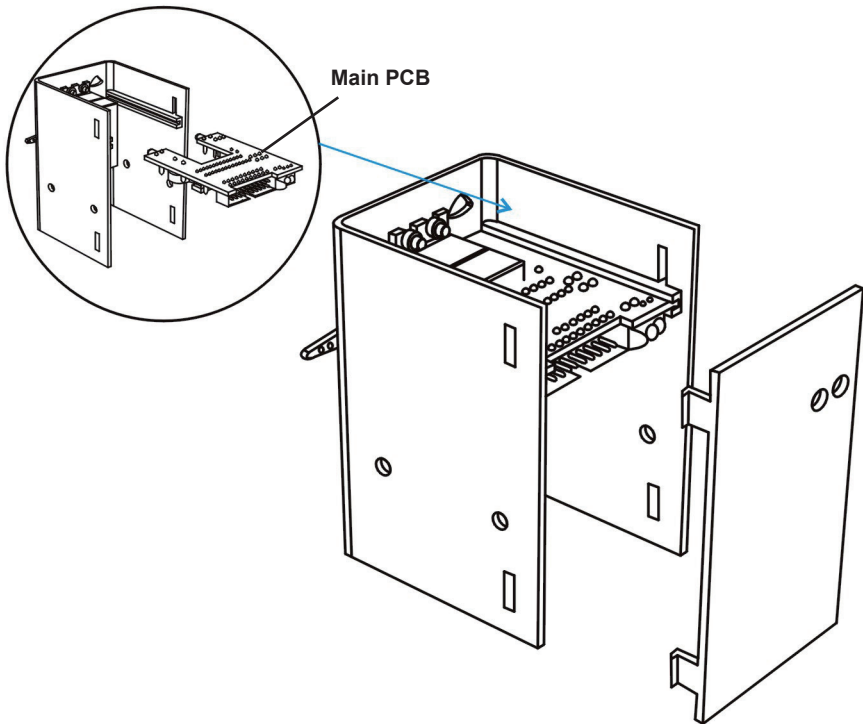


Assemble the bottom plate as described in the drawings.

Assembly YETI head:

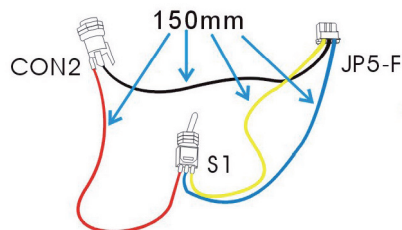
For this final assembly you need;

- 1 pc. Assembled bottom panel
- 1 pc. Assembled head panel
- 1 pc. Assembled main PCB
- 1 pc. Back panel



IMPORTANT!

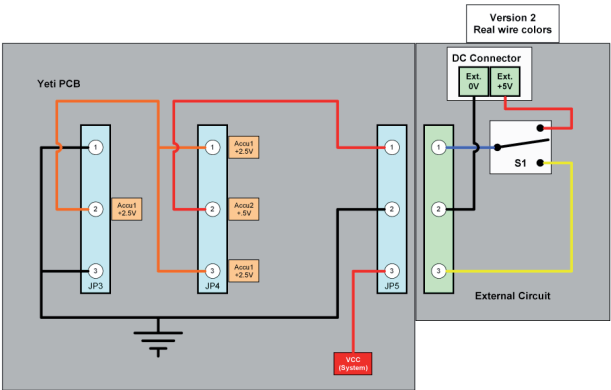
FIRST, before closing the head, you have to install all the wiring and cable sets! See page 33 and 34



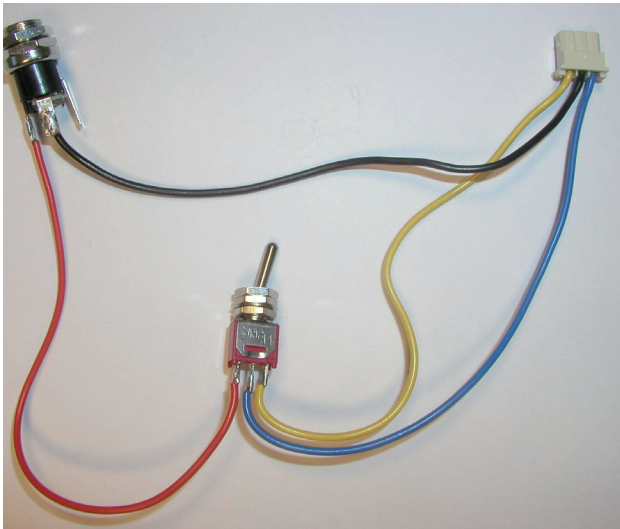
Installing the wires:

For the wire assembly you need;

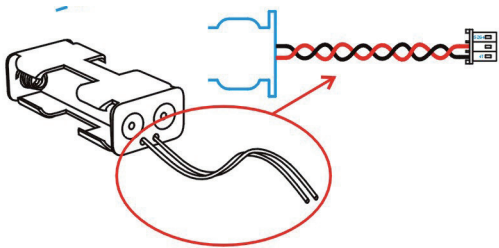
- 1 pc. Assembled bottom panel
- 1 pc. Assembled head panel
- 1 pc. Assembled main PCB
- Assembled cable set with:
 - 1 pc. Wire back
 - 1 pc. Wire blue
 - 1 pc. Wire yellow
 - 1 pc. Wire red



Install all wiring as shown in the drawings.

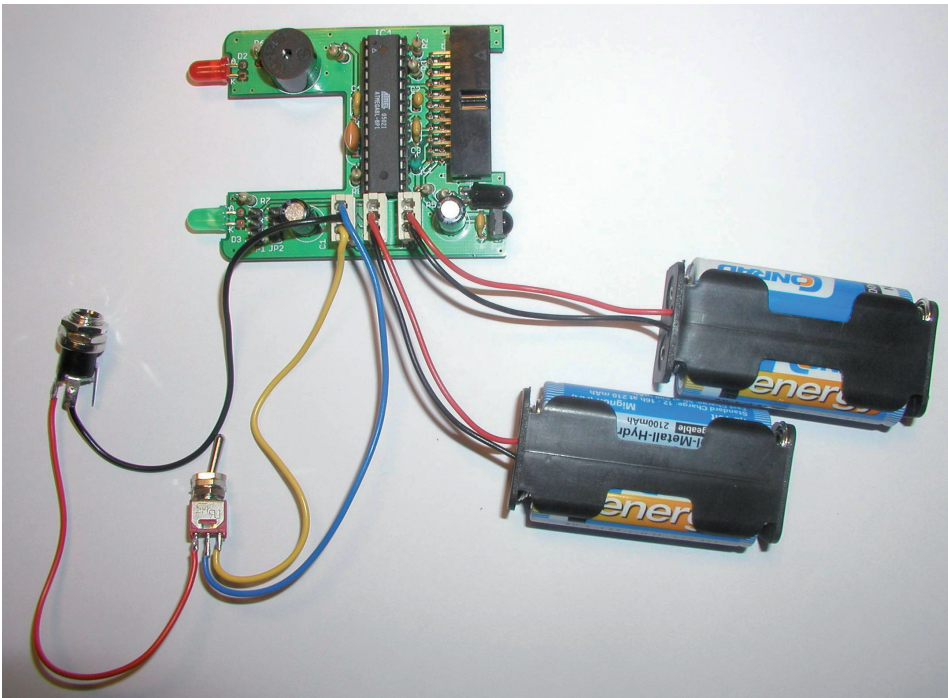
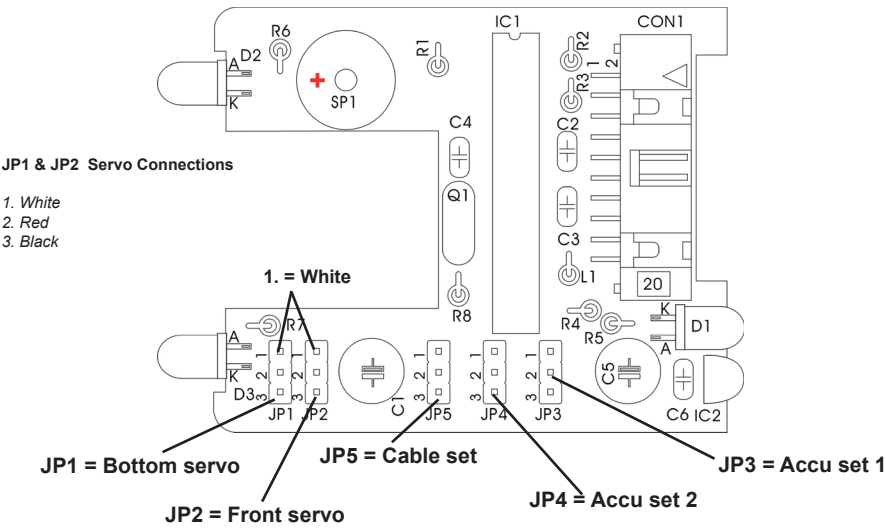


Assembled wire set
Normally pre-assembled



Assembled
battery holder

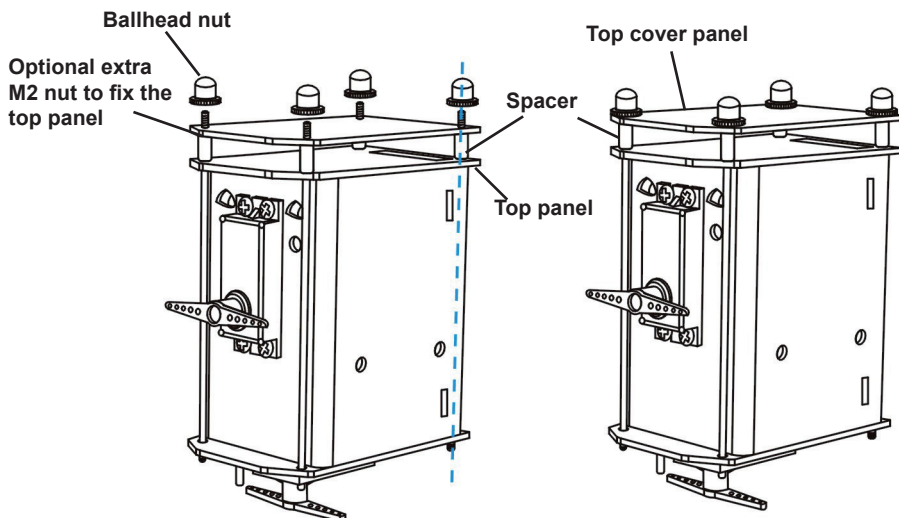
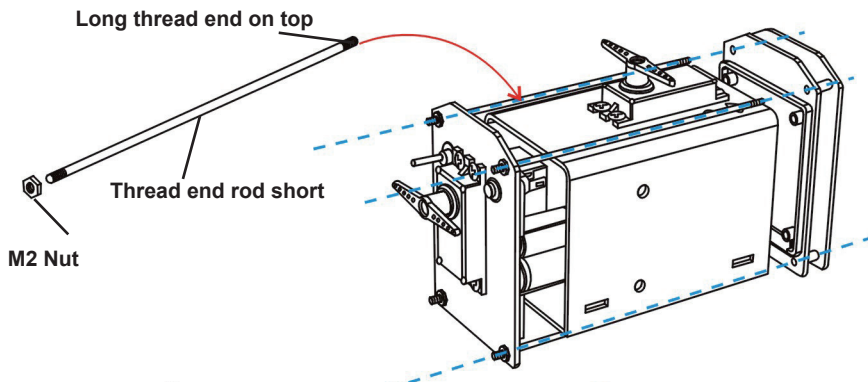
Cable connections main PCB:



Mechanical end assembly YETI head:

For this final head assembly you need;

- 1 pc. Assembled YETI head
- 1 pc. Top panel
- 1 pc. Top cover panel
- 4 pcs. Thread end rod, short
- 4 pcs. M2 Nut
- 4 pcs. Spacer
- 4 pcs. Ballhead nut

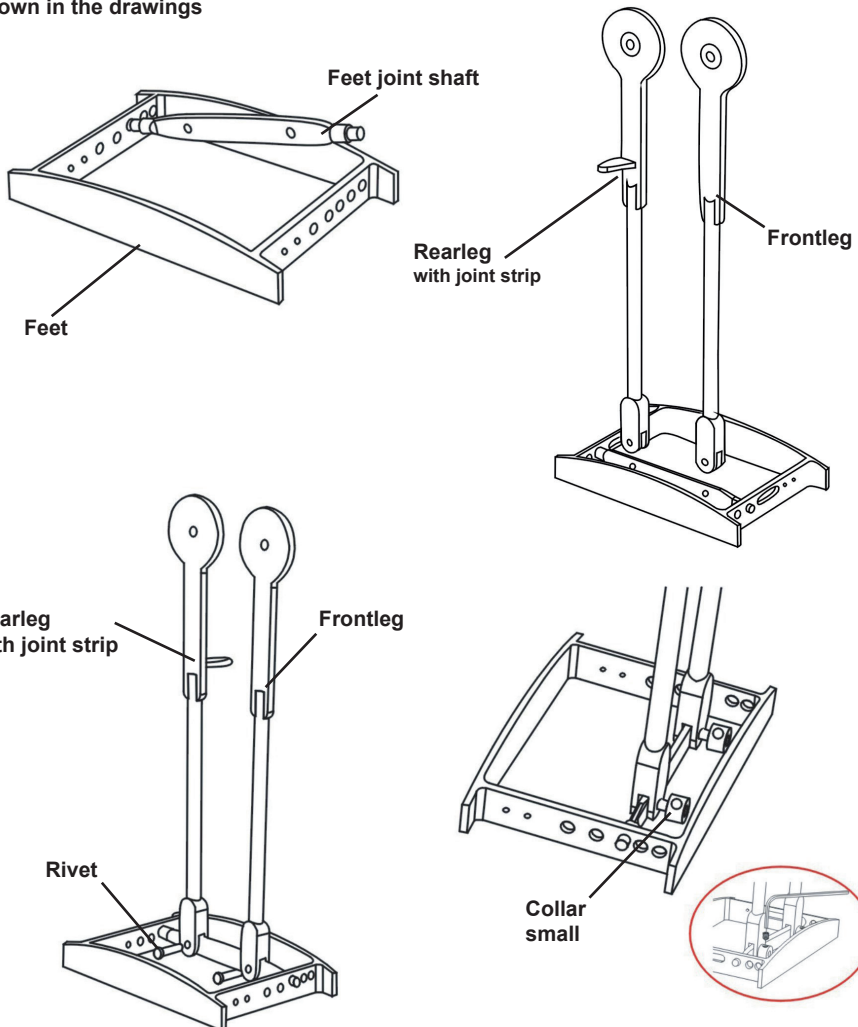


Legs and Feet assembly:

For the legs and feet assembly you need;

2 pcs. Foot
2 pcs. Front leg
2 pcs. Back leg
2 pcs. Feet joint shaft
2 pcs. Rivet
2 pcs. Collar, small

Assemble the feet and legs exactly as shown in the drawings

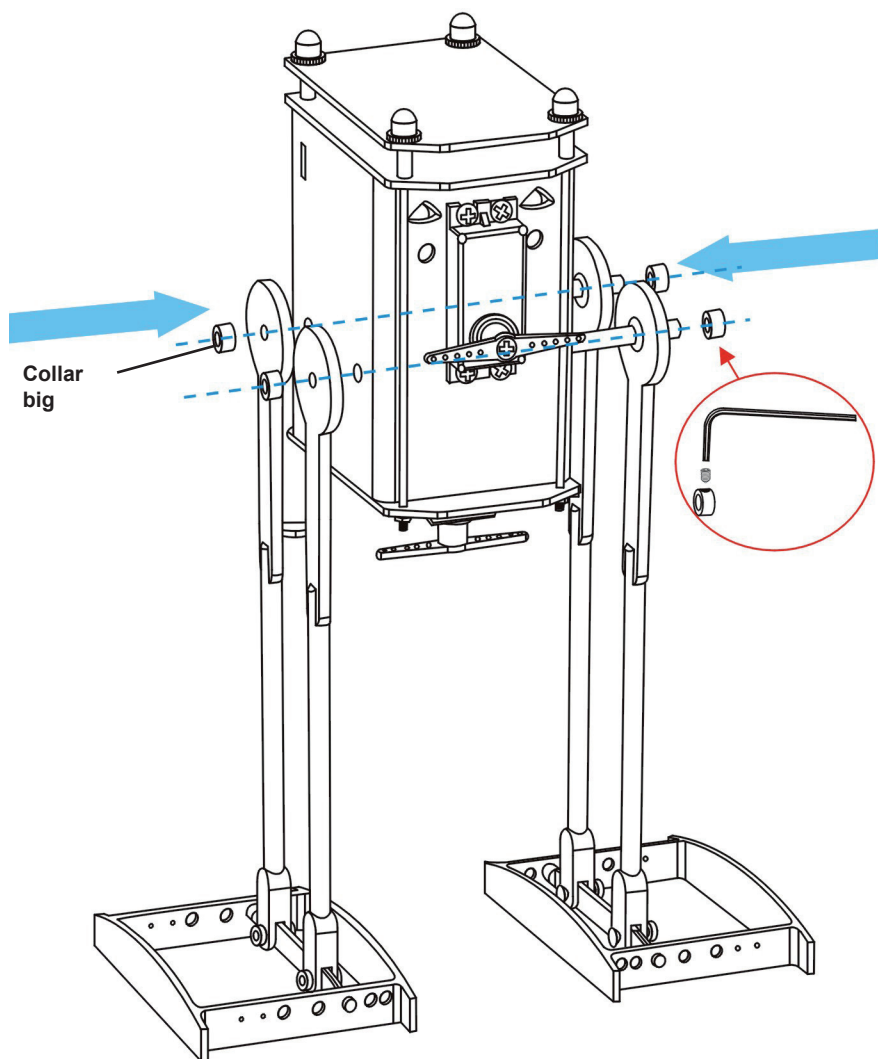


YETI Leg Assembly part I:

For part I of the leg assembly you need;

- 1 pc. Assembled Chassis
- 1 pc. Assembled legs right
- 1 pc. Assembled legs left
- 2 pcs. Rod 5 x 80mm
- 4 pcs. Collar big

Assemble the legs as shown in the drawings

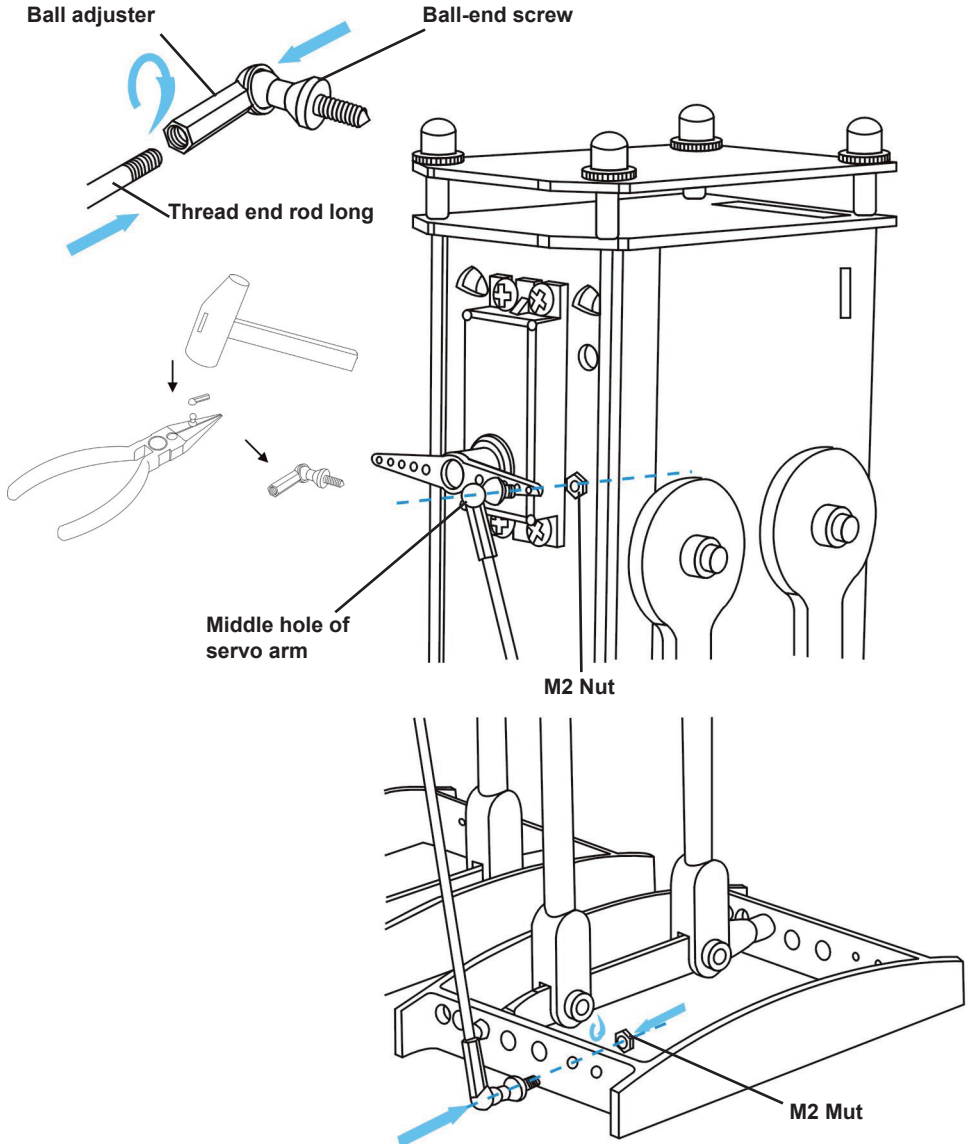


YETI Leg Assembly part II:

For part II of the leg assembly you need;

- 1 pc. Assembled Chassis
- 4 pcs. Ball-end screw
- 4 pcs. Ball adjuster
- 2 pcs. Thread end rod long
- 4 pcs. Nut M2

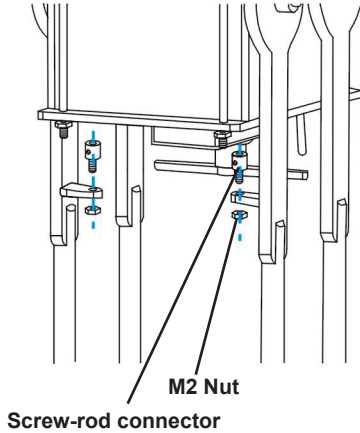
Assemble the servo rods exactly as shown in the drawings



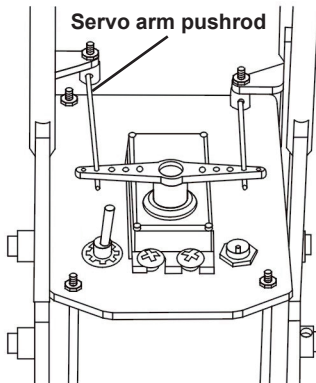
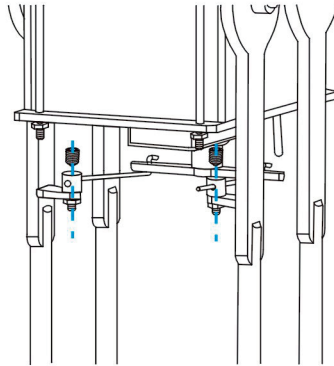
Final assembly YETI legs:

For the final leg assembly you need;

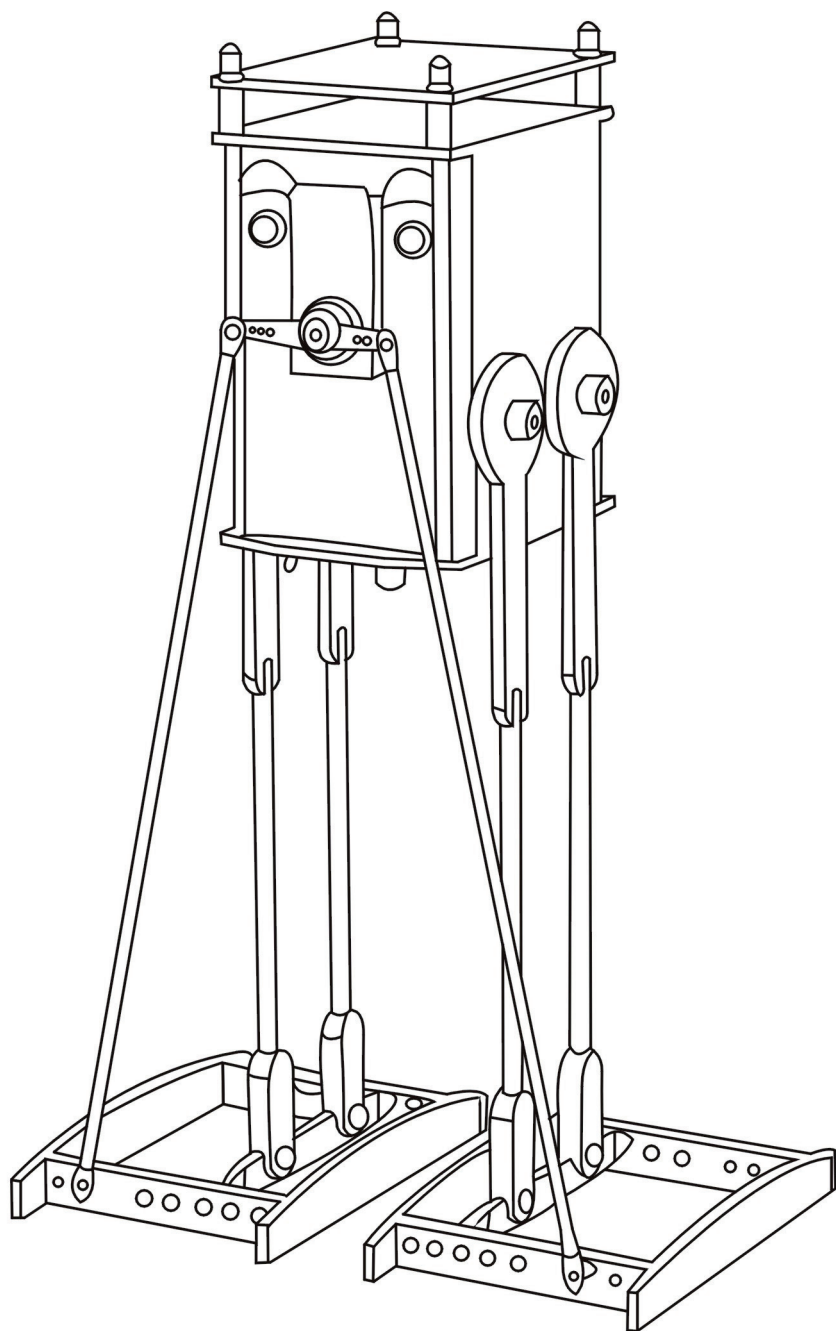
- 1 pc. Assembled Chassis
- 2 pcs. Nut M2
- 2 pcs. Servo arm pushrod
- 2 pcs. Screw-rod connector



Assemble the servo to the rod as shown on the drawings



YOUR YETI IS READY !



8. CHARGING THE YETI BATTERIES

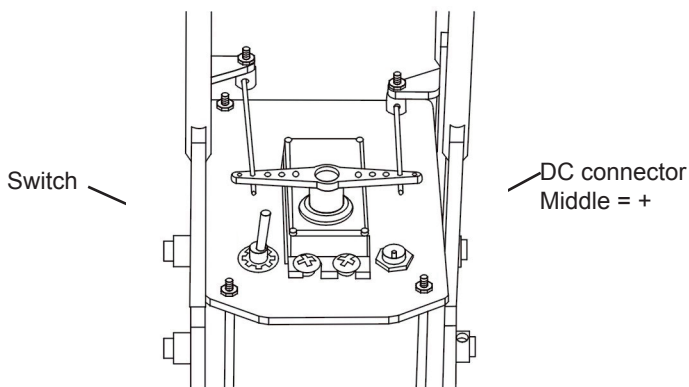
The YETI power voltage is 4.8 Volt supplied by 4 NiCd (1,2V) batteries.

IMPORTANT:

The batteries are not protected by a fuse or series resistor!

Be sure that you are using a good quality battery charger. Best is a microprocessor controlled type!

Or use a stabilized power adaptor with a very small loading current, for example 5 Volt / 300 mA.



Charging YETI batteries:

1. Connect the charger to the DC connector.
2. Put the switch in the off position!

See also APPENDIX K

9. SOFTWARE

9.1. Designing and writing your own YETI program

The following chapter will give all non-experienced programmers a hand and a little help, providing some background information and details in the field of programming.

The chapter is rather difficult and is containing a number of new definitions and words, but will certainly have a positive effect. A common basic knowledge will help you in asking questions, reading documents from the Internet user clubs and asking experts for help or information.

Of course you will be able to load a self-written program into the YETI robot, but how?

- You write a program in a language called "C" (example given: "test.c")
- You compile the program, converting it into a hex file (example given: "test.hex")
- Transfer the hex file to the YETI

You will need just three steps to write the program. In fact experienced programmers are using this simple procedure regularly. But we first need to help the beginners by explaining the method step by step.

9.2. Step 1 Writing a “C”-program

Normally you will be designing and writing a YETI program for a dedicated programming language. For this purpose we choose a very popular programming language called “C”. You will need a special word processor (an editor) to write the YETI program. The preferred editor is called “Programmers Notepad 2” (PN2).

Of course you might also be able to write programs by using “Notepad” or a standard word processor like MS Word but we strongly advise not to use a word processor for this, because the editing procedure will be much more complicated.

In fact the PN2-program is providing some features to help you in editing and trouble shooting in various programming languages. You will just need to select the language, e.g. “C”, “Visual Basic” or “HTML” in your PN2-setup and the PN2-editor will help you by highlighting all special commands, comments, functions and variables in various colors. These features will be a great help in programming.

PN2 also allows you to add self-defined commands to the menu structure. A standard word processor will not allow you to add these commands. In any case the first step in writing your own program will result in a simple text file.

9.3. Step 2 Compiling a “C”-program

A self-designed YETI program, also being called a “source” or a “source program”, basically is a simple text document, example given the file: “test.c”.

Executing a so called “compiler”, which is called GCC.exe in our system, will transform our source code “test.c” into an object file, called “test.o”. An object file is a readable text file, containing a set of “assembler”-instructions. The YETI will be able to understand and execute this list of “assembler”-instructions, but first they need to be processed into another code by a final step.

An “.o” (object)-file obviously contains your “C”-source code and additionally a list of processor instructions (for the specific processor type you will be using).

In a final step, a so-called “linker” will transform these instructions into the processor’s executable instructions (in hexadecimal code), which will be stored in a hex-file, e.g. “test.hex”.

Basically any program modification will have to be followed by a compiler and linker process, but we are able to simplify things by using automatic batch processing and using a “makefile”-process.

Batch processing allows you to chain a great number of different jobs and serialize the process. A makefile allows you how and which files are to be compiled and linked. See the website

<http://www.gnu.org/software/make> for help in using batch jobs and

<http://www.gnu.org/software/make/manual> for help in using makefiles.

These methods imply a great number of new file types and file names, which seems to be complicated and difficult, but working in a standard developing system we normally may ignore these details and leave them to the experts. We will simplify the process radically.

How do we simplify things ... Well, we just start a batch job, which in turn calls a prepared makefile. Starting the job by pressing a button in the standard PN2-menu will allow us to provide an automatic the compiler and linker process in our editor system.

Compiling and linking now turns out to be reduced to activating a button in a PN2-menu or activating a special key in the PN2-keyboard. A one-click-solution!

Imagine a great number of jobs, being processed in a background queue and resulting in a HEX-file, to be loaded in the YETI-processor.

The PN2-editor is a universal programming system, designed to be used for a great number of different compiler and linker systems, which results in a complex interface between the PN2-program and the compiler- / linker-programs.

This looks somewhat complicated. Well, in fact it is rather complicated. But there is no need to learn these complex methods to start writing "C"-programs.

The interface and its configuration will be explained step by step in this manual.

Why did we choose to use this GCC compiler anyway?

The main advantage is the “gnu”-public license and high quality level of the GCC compiler system, which is developed and maintained by the “gnu”-organisation. See: <http://www.gnu.org/>

But the MOST important reason to choose GCC is, because it is free open source software!

Some enthusiast programmers developed a dedicated free software package, named WINAVR (pronounced as whenever), especially designed to program the Atmega processor types, which are also being used in the YETI robot.

The WINAVR package contains the compiler program AVR-GCC, the linker, PN2 and a great number of microprocessor files. The WINAVR software is delivered as a single installation file, which may be found on the CD, included in the robot set. The latest version however may also directly be downloaded from the website:

<http://www.sourceforge.net/> (and search for WINAVR)

The installation procedure creates hundreds of files, from which only a few files will be used.

9.4. Step 3 Transferring a YETI-program

DLR (German Centre for Aerospace and Space Technology) developed a tool called Flash, which has been developed to transfer a YETI-program to the YETI robot by the dedicated COM-port, included in the robot set. The Flash tool takes care of all actions needed to download the selected YETI HEX program, e.g. "test.hex", from the PC into the YETI microprocessor.

The YETI contains a small built-in, hidden communication tool, called "bootloader", which cannot be accessed or deleted by user actions.

Switching on the YETI will always start the bootloader first. The hidden communication tool listens for 3 seconds, waiting for infrared signals from the computer's Flash program. On reception of a Flash signal the YETI starts reception of the program and writes the program into its memory. If no signal is found, the YETI will proceed by processing the program, which previously had been loaded in the memory.

YETI is equipped with a special memory type preserving data after switching off the robot. This type of memory is called "Flash"-memory and of course "Flash" may also be a suitable name for an uploading tool for this type of memory.

10. SOFTWARE INSTALLATION AND INITIAL STEPS

Insert the YETI CD. When all is OK it will start automatically, otherwise open it with windows explorer. After the language selection you will find all the programs you need for the YETI under the software menu. Before you can work with them you have to install the program on your hard drive first. To install programs on your hard drive, you need administrator rights. When you are not logged in as administrator, log out and log on again as administrator.

During the software installation the following will take place:

1. Flash-Tool, a program to transmit your own program to YETI, will be installed.
2. A program editor (Programmers Notepad 2, PN2) and a Compiler (WinAVR) will be installed.
3. An example program(s) will be copied from CDRom to your hard disk.
4. In program editor (PN2), a menu input for Make and Clean files, will be created.

10.1 Windows

10.1.1 Flash-Tool

Copy the flash program in a folder on your hard drive for example C:\programs\flash.

It is also possible to run the flash program directly from the CD.

In all cases, it is wise to make a link for the flash program on your desktop.

10.1.2 Installation of the program editor and compiler

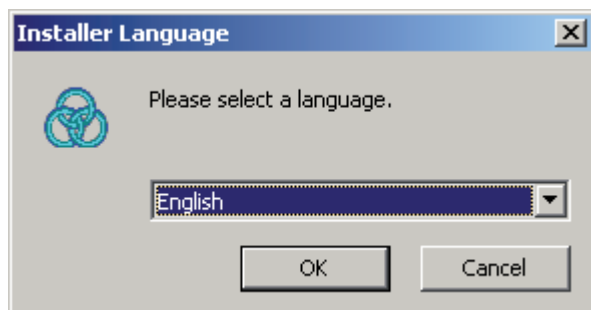
For the installation of the compiler you must be the administrator of your computer (during the installation the registry will be changed). When you are not registered as the administrator, restart the computer and start up as administrator!

Click on the install symbol to install.

COMPILER WinAVR (20060421)

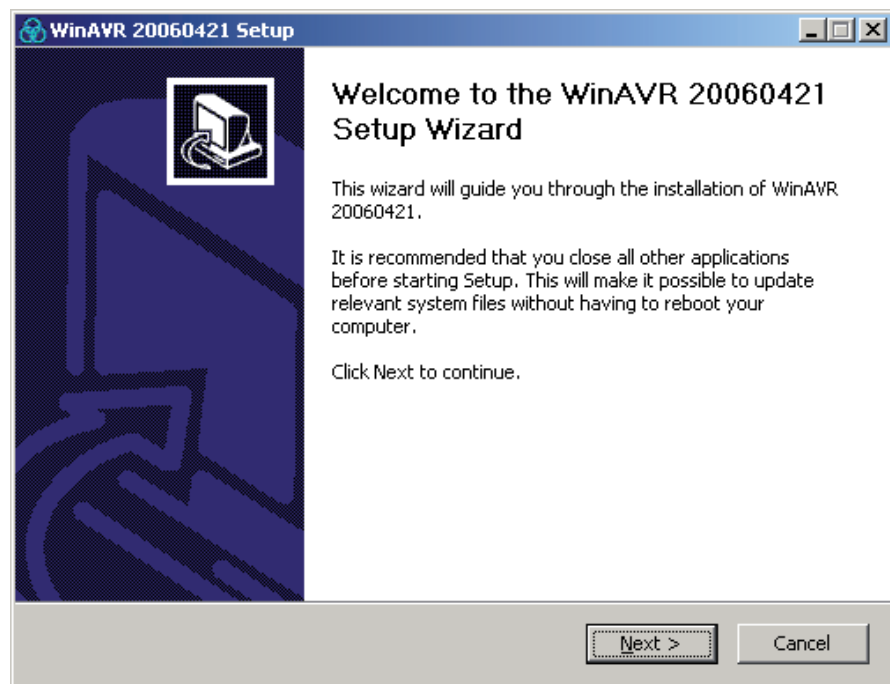
WINDOWS is the trade mark of Microsoft Corporation.

Now this screen will appear:



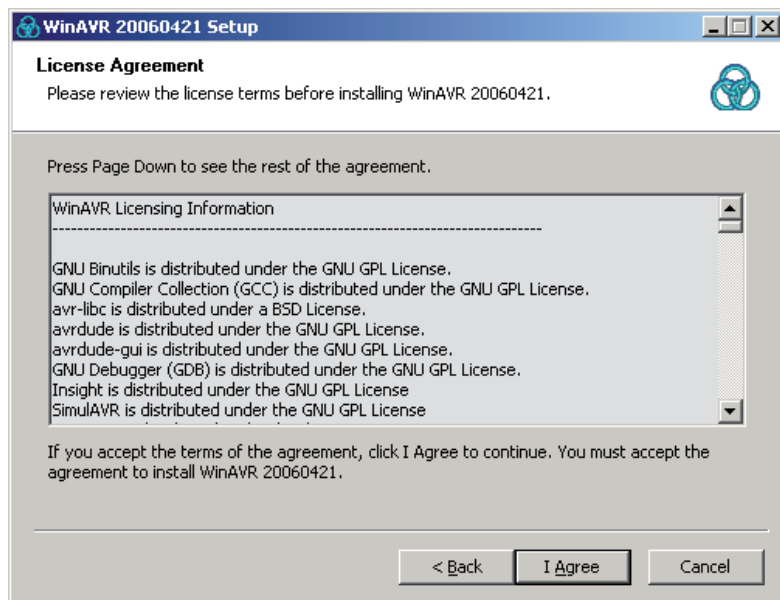
Click OK [I Agree]

This screen appears:



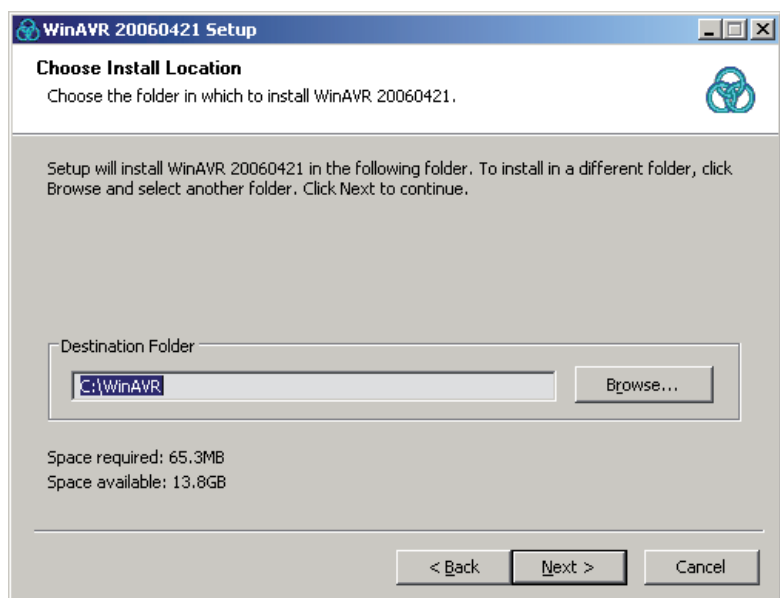
Click continue [Next]

This screen appears:



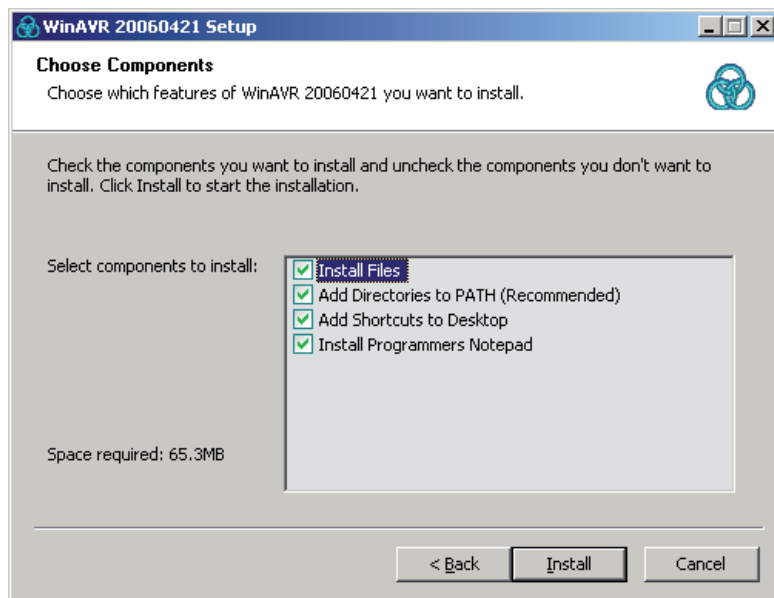
Click accept [Agree]

This screen appears:



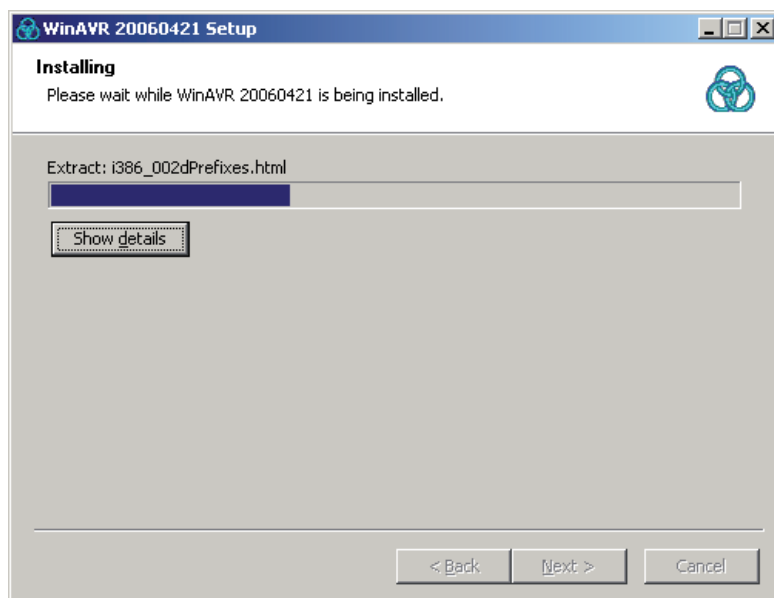
Click Continue [Next]

This screen appears:



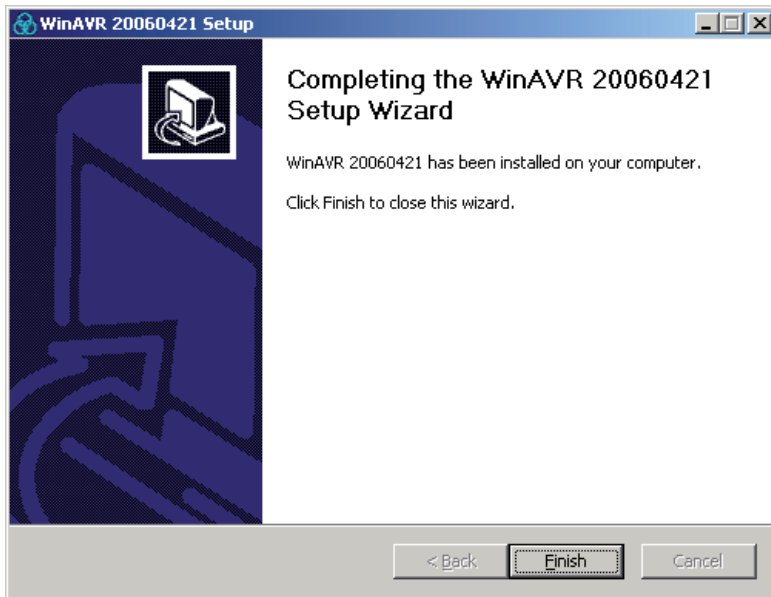
Click Install [install]

This screen appears:



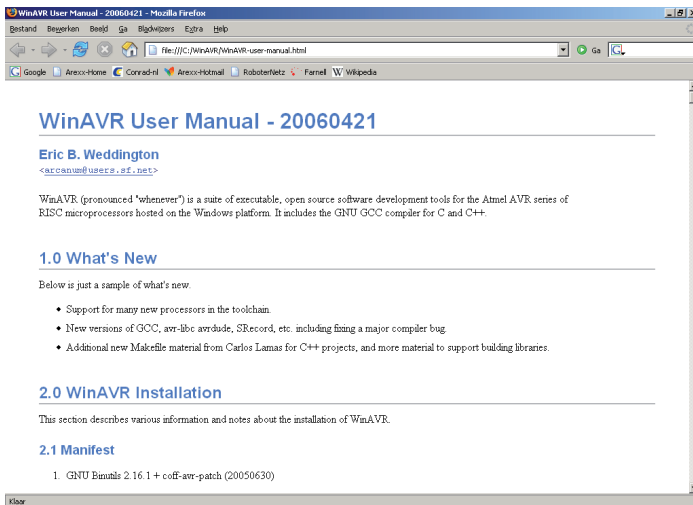
Wait

This screen appears:



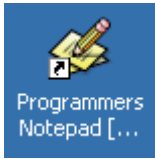
Click safe [safe]

This screen appears:



Close this screen

On the DESKTOP the 'programmers notepad 2' Symbol appears:



The program editor and the compiler are installed now.

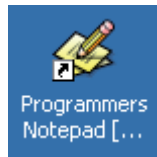
10.1.3. Copying the example programs from CD to the harddisk.

Copy the folder 'YETI_src' from the YETI CD to the hard disk (put it in a folder something like this: 'C:\YETI_src').

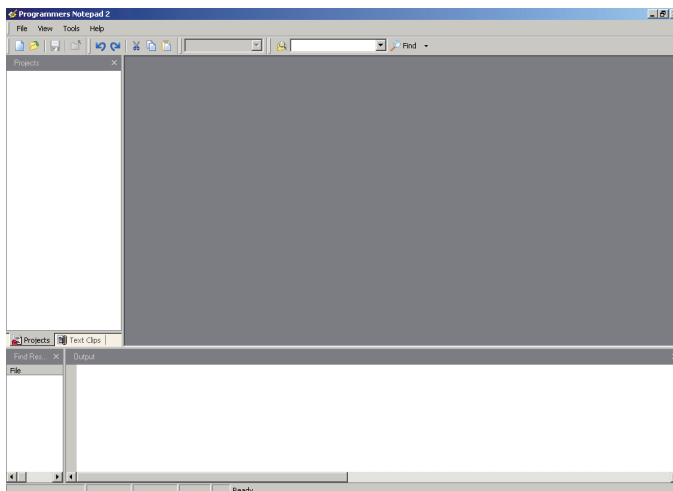
10.1.4. Compiling a 'C' file

Just for try we will open the file 'C:\YETI_src\FirstTry\test.c' :

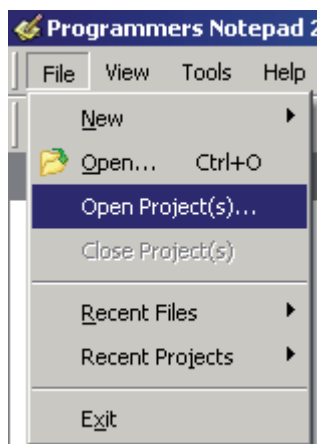
Open Programmers Notepad 2' (click the notepad symbol on your desktop twice):



This screen appears:



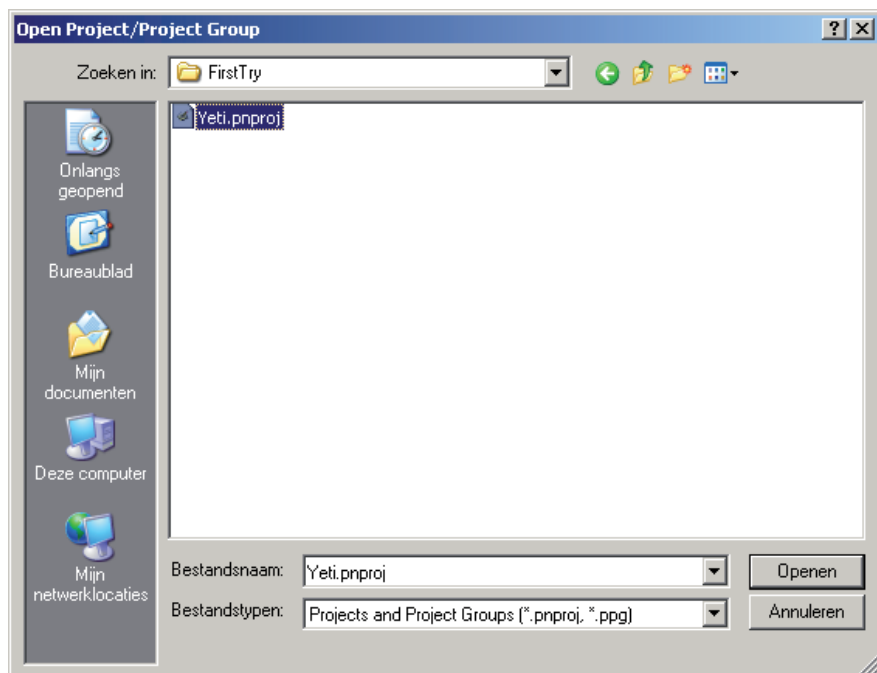
Select: Open Project(s)...



Find the data:

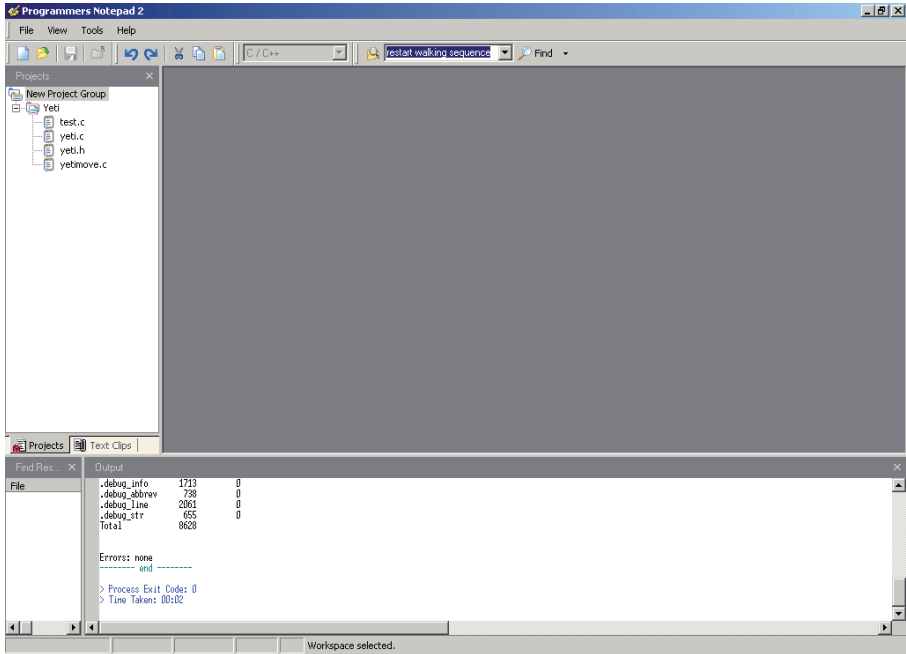
C:\YETI_scr\Firsttry\YETI.pnproj

This screen appears:

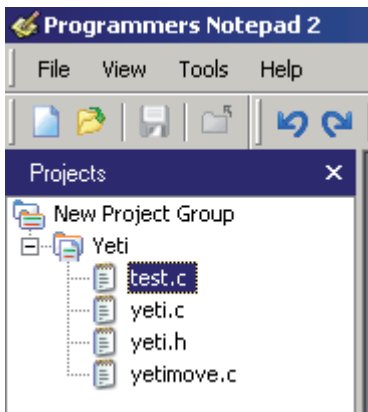


Click Open

This screen appears:



Doubleclick 'test.c':



Programmers Notepad 2 - [test.c]

File Edit View Tools Window Help

Projects

New Project Group

- YETI
 - test.c
 - yeti.c
 - yeti.h
 - yetrove.c

```

//=====
//
// # File Name : test.c
// #
// # Project : YETI
// #
// # Description: This file is the main file for all Yeti applications.
// #
// # (c) Hesk van Winkoop (Openware software development)
// #
// # Version:
// #
// # 1.00      2006-05-04   Original release
// #
//=====

//
//=====
//
// # DEFINES
//
//=====

#define COMPILER_FOR_CALIBRATION

//define COMPILER_FOR_CALIBRATION
//> What is the meaning of "define COMPILER_FOR_CALIBRATION"?
//> Remove the leading "/" in front of "define" and when compiling
//> all program parts starting with
//> "define COMPILER_FOR_CALIBRATION"
//> will now be compiled. Now the program is compiled for the
//> YETI server calibration.
//
//=====

```

Projects | Test Clips | test.c

Find Res. | Output

File

[11]: 172

AVSIG | C:\A\F | JNS | Project file: c:\pwt_uc\frsby\test.c

Programmers Notepad 2 - [test.c]

File Edit View Tools Window Help

Line Endings

☒ Use Tabs

[WinAVR] Make All

[WinAVR] Make Clean

[WinAVR] Program

Stop Tools Ctrl+Shift+K

Options

Projects

New Project Group

Yeti

- test.c
- yeti.c
- yeti.h
- yetimove.c

The screenshot shows the Visual Studio IDE with the Output window open. The Output window displays the following debug statistics:

Category	Value	Count
.debug_info	1713	0
.debug_abbrev	738	0
.debug_line	2061	0
.debug_str	655	0
Total	8628	

Below the statistics, the Output window shows the message "Errors: none" followed by a dashed line and the word "end". At the bottom of the Output window, the following information is displayed:

- > Process Exit Code: 0
- > Time Taken: 00:01

The status bar at the bottom of the IDE shows the current file is "test.c" and the project file is "c:\yeti_src\firsttry\test.c".

Now the compiled data is ready and a file 'test.hex' is generated.

...and when the program does not contain any mistakes (what can be expected, because we just loaded an example program), on the bottom this message will appear: Errors: none.

What happened?

From the file test.c (and YETI.c) a new file test.hex was generated. This file contains the in machine code converted program. This machine code program can be loaded (flashed) in YETI's memory. This program does not have a function. Later on, for trial, we will upload it in the YETI memory with the Flash tool.

How did it work?

The menu input file calls the batch file Test-all.bat (this batch file contains a list with command lines which are executed line after line).

In Test-all.bat the command 'make all' will be executed. 'make' will create a make file which will be located (when we program YETI) in the same file as Test-all.bat.

A make file is a text file, which defines how to compile one or more programs. During programming, when it is only one file, you still will have a good overview. Later, when a complex system is written and the programming data contains more files, which all must be converted in the correct way step by step and also connected (linked) with each other in a proper way, then also the make file will be very complex.

The "all" calls for all the input in the make file means, that a complete project and not only the separate inputs will be converted.

The make file in our example program is written in a way that a file with the name test.c will be compiled together with YETI.c (which contains some pre defined functions) and create a new .hex-file. This file can be loaded (flashed) into the YETI memory.

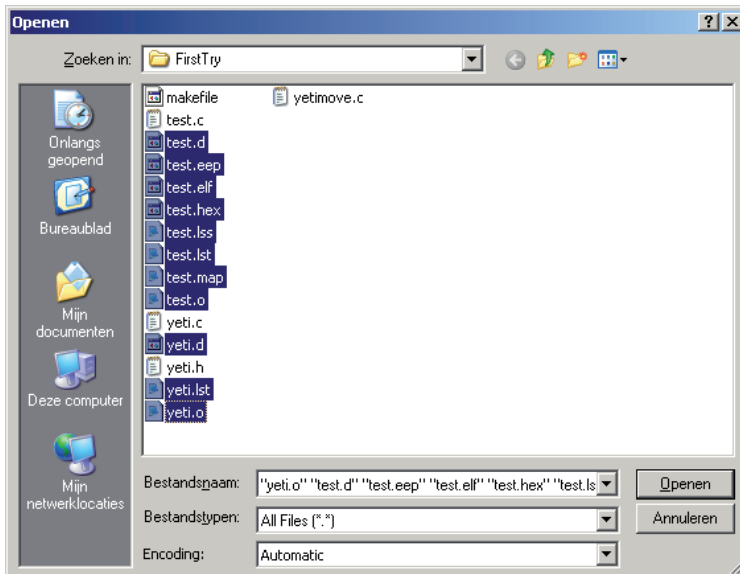
Attention! This means that - as long as you do not change the make file and you only copy it - you should always name your own program test.c .

When you want to know all about make files (This is not absolutely necessary for operating the YETI) you can find more background information at <http://www.gnu.org>

The basics about ASURO-programming will be explained in chapter 9.

When you compile a program, some extra data is generated. This data is only necessary during the conversion, after that it is useless. These data files can be removed with the clean tool.

With 'File -> Open' you can see the generated data. We have marked the new data. You can see new data e.g. 'test.hex'



With 'Tools -> [WinAVR] Make Clean' you can delete the marked data.

READY

MORE WINAVR information see;

<http://www.avrfreaks.net>

<http://winavr.sourceforge.net>

<http://www.kreatives-chaos.com> --> software

10.2. LINUX

For the software installation you need root rights. If you do not have these, log out and log in again as root or open a shell and demand the root with 'su'.

10.2.1 Flash-Tool

Start the program from the CD software menu and copy the two flash tools "yetiflash" and "yeticon" from the folder "/linux/tools" into the folder "usr/local/bin". After this, you must allow the execution of the program with "chmod a+x /usr/local/bin yeticon yetiflash"

Wird ein in einer Shell eingetipptes "yetiflash" nicht gefunden, muss der Pfad "/usr/local/bin" noch der %PATH-Variable hinzugefügt oder das Programm mit vollem Pfad aufgerufen werden.

10.2.2 Compiler

To install the Gnu-Compilers for AVR-Processors, insert the YETI CD-ROM and choose the following from the folder “/Linux/Compiler” :

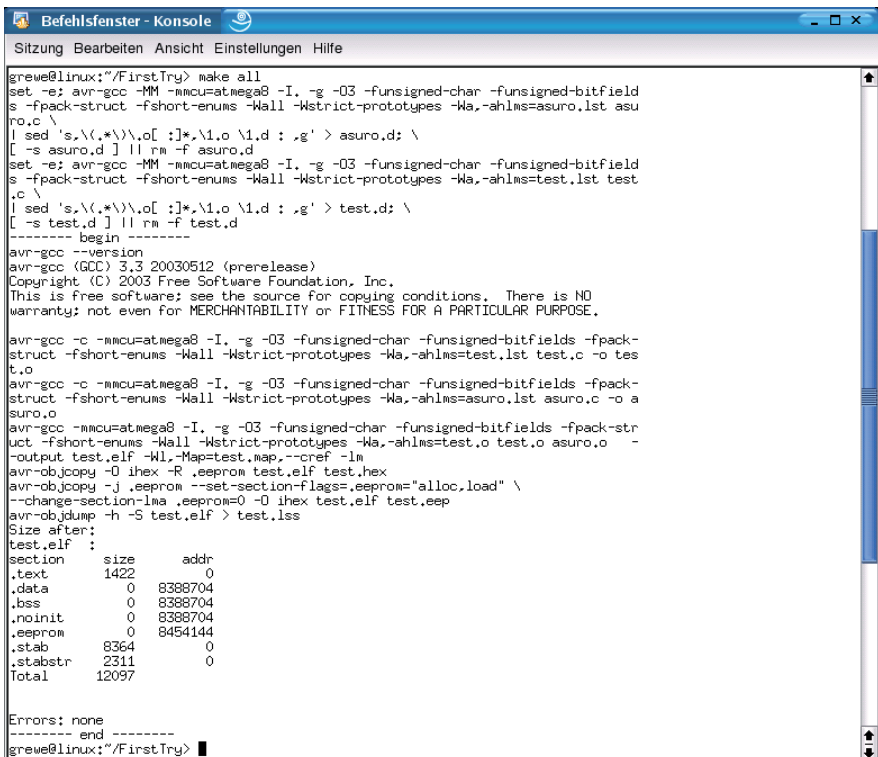
1. **avr-binutils-... .rpm**
2. **avr-gcc-... .rpm**
3. **avr-libc-... .rpm**

The installation is quite simple!

Just give the command: `rpm -i <paket>.rpm` in your root directory.

Ready!

For Editors you can use Exmacs, Kate or Kedit. For trial you can copy the demo programs from the CD. You can find these in the folder “/YETI_src/FirstTry/”. Then you can open a Shell, change the folder and enter “make”. When all is OK you will see the following window; (see fig. 10.1)



```
grewe@linux:~/FirstTry> make all
set -e; avr-gcc -MM -mmcu=atmega8 -I. -g -O3 -funsigned-char -funsigned-bitfield
s -fpack-struct -fshort-enums -Wall -Wstrict-prototypes -Wa,-ahls=asuro.lst asu
ro.c \
| sed 's,\(.*\)\\.o[ :]*,\1.o \1.d : ,g' > asuro.d; \
[ -s asuro.d ] || rm -f asuro.d
set -e; avr-gcc -MM -mmcu=atmega8 -I. -g -O3 -funsigned-char -funsigned-bitfield
s -fpack-struct -fshort-enums -Wall -Wstrict-prototypes -Wa,-ahls=test.lst tes
t.c \
| sed 's,\(.*\)\\.o[ :]*,\1.o \1.d : ,g' > test.d; \
[ -s test.d ] || rm -f test.d
----- begin -----
avr-gcc --version
avr-gcc (GCC) 3.3 20030512 (prerelease)
Copyright (C) 2003 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

avr-gcc -c -mmcu=atmega8 -I. -g -O3 -funsigned-char -funsigned-bitfields -fpack-
struct -fshort-enums -Wall -Wstrict-prototypes -Wa,-ahls=test.lst test.c -o tes
t.o
avr-gcc -c -mmcu=atmega8 -I. -g -O3 -funsigned-char -funsigned-bitfields -fpack-
struct -fshort-enums -Wall -Wstrict-prototypes -Wa,-ahls=asuro.lst asuro.c -o a
suro.o
avr-gcc -mmcu=atmega8 -I. -g -O3 -funsigned-char -funsigned-bitfields -fpack-str
uct -fshort-enums -Wall -Wstrict-prototypes -Wa,-ahls=test.o test.o asuro.o -
-output test.elf -Wl,-Map=test.map,--cref -lm
avr-objcopy -O ihex -R .eeprom test.elf test.hex
avr-objcopy -j .eeprom --set-section-flags=.eeprom="alloc,load" \
--change-section-lma .eeprom=0 -O ihex test.elf test.eep
avr-objdump -h -S test.elf > test.lss
Size after:
test.elf :
section      size      addr
.text        1422        0
.data         0      8398704
.bss          0      8398704
.noinit       0      8398704
.eeprom       0      8454144
.stab        8364        0
.stabstr     2311        0
Total       12097

Errors: none
----- end -----
grewe@linux:~/FirstTry> █
```

Fig 10.1.: Make all

10.3. Flash - The YETI-programming-tool

In this step we will need the Flash program (see fig. 10.2).

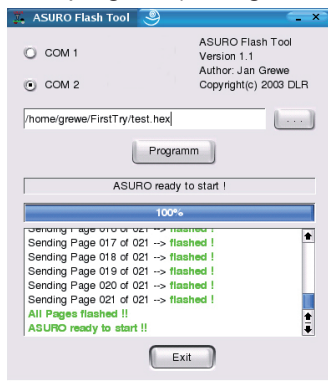
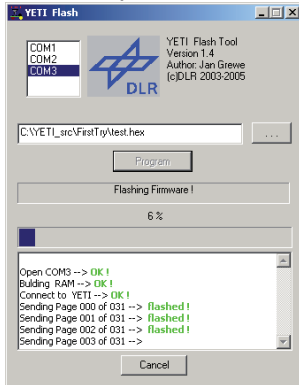


Fig. 10.2.:
Flash-Tools for
Windows and LINUX

Start the program and select the interface in which you have plugged the IR-Transceiver. Select Test.hex from directory C:\Own files\YETI_src\FirstTry.

Place the completely assembled and tested YETI near the IR-Transceiver, at a distance of max. 50 cm. The component sides of both PCBs must be facing and “seeing” each other. Click the Program button at the Flash-Tool. Now switch S1 to ON-position, before the status-indicator reaches the right end of the status-area. If you have failed to react fast enough or communication has been disturbed, just switch YETI off, press Programm and switch S1 to ON-position again.

As soon as communication is succesful, you may observe in the status-indicator and display how the file Test.hex is being transferred to YETI. The program file will be stored in the Flash-memory inside the processor, where the program remains available even after switching off the supply voltage. After loading the program, YETI will have to be switched OFF and ON again in order to start the program. This sequence will execute the loaded program and the green LED will lit up brightly.

10.3.1. What is happening while flashing?

As soon as the Flash program has been executed, the PC will try to communicate with YETI. By switching on YETI, the system will be booted, indicated by the Status-LED lighting for one second. YETI is checking to see, if new software has been prepared to be loaded. If a new program has been found, it will be loaded. After loading, the program can be executed by switching OFF and ON once again.

10.4. Flash failures

The following errors may occur while flashing:

- "C" Checksum Error. YETI has received some irregular signals. Signals may have been disturbed by other optical sources, such as fluorescent lights, or have been interrupted shortly by movements.
- "t" Timeout. The line-of-sight between YETI and the IR-Transceiver has been interrupted completely.
- "V" Verify Error. YETI wrote invalid data into its Flash-memory. This is a most unusual situation, indicating the non-volatile programming memory (Flash-EEPROM) has reached the end of its lifetime, according to specifications after approximately 10.000 programming cycles.

Error correction can be retried ten times. In case of failure the flash-procedure will be aborted.



If Checksum Errors are being indicated regularly while Flashing, you may switch off or dim some lights in the room, especially fluorescent lights.



Remember:

*Always press the Program-button, before switching ON the YETI.
Otherwise the download procedure will not be started.*

11. TEST AND OPERATION

After completing the assembly, we will start moving the robot. But first we have to find and eliminate the errors we could have made in the previous phase, without destroying any parts.

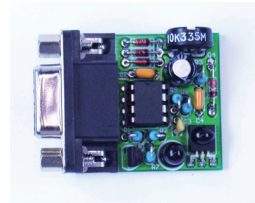
11.1. RS232-Infrared-Transceiver

The following operational check is limited to the RS232-Infrared-Transceiver. First of all the IR-Transceiver must be checked, as it will be needed for the next step: the selftest of the system. For this test connect the IR-Transceiver to a free serial interface of your PC with the supplied 9-pole serial extension cable and start the Windows terminal program "Hyperterminal" (or for a Linux-system eg. "Minicom"). Normally you will find this program in Windows in the category Addons --> Communication --> Hyperterminal. If the program is not available, you can install it from the Windows-CD.

After starting the Hyperterminal program you will be asked to define a name for the connection. You may choose YETI or any other symbol. In the next window you choose "connect by" and the COM-interface by which the transceiver has been connected in the previous step.

Then press "OK" and choose the following settings:

- Bits per Second: 2400
- Databits: 8
- Parity: none
- Stopbits: 1
- Flowcontrol: none



Press "OK" again for confirmation

Hold the IR-Transceiver at the distance of 10 cm over a white sheet of paper. The component side must be directed towards the paper sheet.

Press a few keys at your computer terminal.

The terminal program normally should display the key-symbols. The IR-Transceiver transmits the key-symbols by IR-Diode (D5), the transmitted signal reflects at the paper surface and is send back to receiver-IC (IC2), from which it is being returned to the computer. If no symbols or wrong symbols are being displayed, you may carefully turn the trimmer between its extreme left and right position. Use a miniature screwdriver and strike a few keys at each position of the trimmer until the correct symbols are displayed.

If you do not have any success in this procedure, we do have a problem with the circuit, which should be solved. (see Appendix H or www.arexx.com [for help](#)).

Just to be sure you should remove the IR-Transceiver after this test and hit a few keys. This time the display is expected to show no symbols any more.

11.2. USB-Infrared-Tranceiver

This operation is only for the USB IR-Transceiver.

WARNING! The USB IR-Transceiver does not have a housing and therefore it is very sensitive for electrostatic discharge. Before you use it, discharge your body by touching a metal computer housing or other earth point. An other option is to build the USB IR-Transceiver in a transparent housing for further protection.

11.2.1 Windows

The following operational check is limited to the USB Infrared-Tranceiver.

First of all the IR-Transceiver must be checked, as it will be needed for the next step: the selftest of the system. For this test connect the IR-Transceiver to a free USB port of your PC by the USB extension cable.

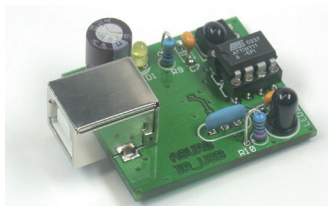
Now a message will appear "NEW HARWARE WAS FOUND";

AREXX ASURO/YETI USB-IR-TRANSCIEVER

Now you can install the USB driver from the YETI CD. When the driver is not detected automatically, you can select it manually from CD\windows\USB Driver (Administrator rights are necessary for this operation). When the driver is installed you can approach the USB Transceiver like a normal serial port. After starting the Hyperterminal program, you will be asked to define a name for the connection. You may choose YETIUSB or any other symbol. In the next window you choose "connect by" and the COM-interface by which the transceiver has been connected in the previous step.

Then press "OK" and choose the following settings:

- Bits pro Second: 2400
- Databits: 8
- Parity: none
- Stopbits: 1
- Flowcontrol: none



Press "OK" again for confirmation

Hold the IR-Transceiver at the distance of 10 cm over a white sheet of paper. The component side must be directed towards the paper sheet. Press a few keys at your computer terminal. The terminal program normally should display the key-symbols. The IR-Transceiver transmits the key-symbols by IR-Diode (D5), the transmitted signal reflects at the paper surface and is send back to receiver-IC (IC2), from which it is being returned to the computer.

If you do not have any success in this procedure, we do have a problem with the circuit, which should be solved. (see Appendix H or www.arexx.com for help). Just to be sure you should remove the IR-Transceiver after this test and hit a few keys. This time the display is expected to show no symbols any more.

11.2.2 Linux

The following operational check is limited to the USB Infrared-Tranceiver and LINUX software. First of all the IR-Transceiver must be checked, as it will be needed for the next step: the selftest of the system. For this test connect the IR-Transceiver to a free USB port of your PC by a USB extension cable. A short beep will confirm that the transceiver was detected by the LINUX software. To be sure please check in the proc-declaration if the following message is displayed.

```
foo@bar:~>cat /proc/tty/driver/usb-serial
```

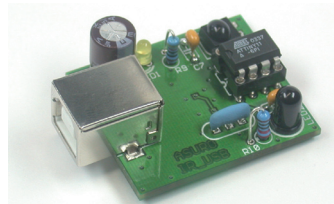
There is also an entry with the following ('0' in our example can also be '1' or '2' etc.):

```
usbserinfo:1.0 driver:v1.4
```

```
0: module:ftdi_sio name:"FTDI 8U232AM Compatible" vendor:0403 product:6001  
num_ports:1 port:1 path:usb-00:11.2-1
```

For the test you have to configure the Minicom at the interface /dev/ttyUSB0 (oder 1, 2 usw...) with the following settings:

- Bits pro Second: 2400
- Databits: 8
- Parity: none
- Stopbits: 1
- Flowcontrol: none



Press "OK" again for confirmation

It is possible that root rights are necessary.

Maybe you need to declare read and write rights for the user or groups for the new device /dev/ttyUSB?. You can do this with `chmod u+r+w /dev/ttyUSB0` (oder 1, 2...) or `chmod g+r+w /dev/ttyUSB0` (of course again you need the root.rights).

Hold the IR-Transceiver at the distance of 10 cm over a white sheet of paper. The component side must be directed towards the paper sheet.

Press a few keys at your computer terminal.

The terminal program normally should display the key-symbols. The IR-Transceiver transmits the key-symbols by IR-Diode (D5), the transmitted signal reflects at the paper surface and is send back to receiver-IC (IC2), from which it is being returned to the computer. If no symbols or wrong symbols are being displayed you may carefully turn the trimmer between its extreme left and right position.

When it does not work 100% OK, please visit www.arexx.com --> Forum or www.roboternetz.de --> Forum.

12. CALIBRATION

For a stable walking process a correct calibration procedure is necessary, which needs to be executed only once. A software calibration is necessary so we can adjust the servomotors to the middle position with our software. A hardware calibration also needs to be executed so we can align the legs and feet in accordance with the middle position of the servomotors

12.1. Software calibration

For a stable walking process a correct calibration procedure is necessary, which needs to be executed only once. To do so we need the program Hyperterminal, for which version 6.3 can be downloaded from the Internet.

Start Hyperterminal. Setup your Hyperterminal program for COM (X), 2400baud, 8-bits, no parity and no hardware control.

Connect the RS232 or USB-adaptor to your computer.

Turn YETI around with his face to the floor in order to allow a visible contact and infrared data communication between the USB-adaptor and YETI's rear head.

Switch on the YETI.

Ca. 3 seconds later a series of beeps may be heard.
Now press any key at the PC terminal within 1 second.

The keys listed in table 1 are now available to calibrate the YETI.

Start by pressing [-]. Both servos will move to their electrical zero position. Now adjust the "fore"-servo-arm to a crosswise position.

Adjust the arm as exactly as possible without altering the servos shaft position. Adjust the "fore"-servo to an exact crosswise position by pressing keys [7] and [9].

Press key [1] to calibrate the system for this position.

As a testing procedure move the servo to another position by pressing key [7] or [9] and press key [8] to return the servo to the previously calibrated mechanical zero position.

SEE. APPENDIX J for more information

Repeat the calibration procedure for the “lower” servo with keys [4], [5], [6] and [2].

The [+]-key has been defined as a simultaneous combination of [8] and [5].

Pressing the [ENTER] will close the calibration procedure.

At restart, theYeti will start by returning to the calibrated mechanical zero position.

IMPORTANT!
In YETI’s standard supplied programs you will find a function ‘vCalibrateServos()’ to do the calibration process with.

SEE ALSO APPENDIX J for more information

YETI calibration keys			
			[-] [+] Return both Servos to their calibrated zero-position (Identical to [8] & [5]) [ENTER] Terminate calibration
[7] ‘Front’ Servomotor to the left	[8] “Front” Servomotor Calibrated position	[9] ‘Front’ Servomotor to the right	
[4] ‘Botom’ Servomotor to the left	[5] ‘Bottom’ Servomotor Calibrated position	[6] “Bottom” Servomotor to the right	
[1]	[2]		

13. YETI PROGRAMMING

What's next?

Now you completed the Yeti and the robot is working fine. Are we ready now?

No, you are not ready yet. You just completed the overture. The real job is still waiting!

Experienced C-programmers may directly proceed writing software. Beginners may feel more comfortable reading the following chapter completely, even if some parts of the story may seem to be an ancient history.

YETI's brains

We will start a short overview in a summary. The main printed circuit board in the YETI contains a miniature computer, usually named 'microcontroller'. The microcontroller is an integrated circuit (abbreviated IC) and you may easily identify this chip as a small, black 28-legged box. Electric wiring connects the microcontroller directly to the red LED-eyes, to the loudspeaker, to the infrared communication system and to the servomotors, controlling YETI's movements.

This is a short summary indeed, but we will proceed with a step by step explanation for beginners. Just relax for a moment. Soon you will be writing your first programming lines...

YETI's RED EYE LEDS

We will start a short overview in a summary. The main printed circuit board in the YETI contains a miniature computer, usually named 'microcontroller'. The microcontroller is an integrated circuit (abbreviated IC) and you may easily identify this chip as a small, black 28-legged box. Electric wiring connects the microcontroller directly to the red LED-eyes, to the loudspeaker, to the infrared communication system and to the servomotors, controlling the YETI's movements.

This is a short summary indeed, but we will proceed with a step by step explanation for beginners. Just relax for a moment. Soon you will be writing your first programming lines...

Servo-motors

We provide the YETI with two special motors, housed in small compartments and containing a few cog wheels and some control electronics. The output axle is provided with a single cog wheel. Electronic engineers call these devices servos. The internal cog wheels make up a gear-system. In fact the gear slows down the number of revolutions and compared to the motor axle the output cog wheel will be turning rather slowly. In fact the output axle will not complete a full rotation, but instead will cover an angle range up to around 220 degrees. The slow speed however implies a considerable torque or force to the output axle.

Microcontrollers and programming

A microcontroller accepts a set of 120 basic instructions and a great amount of combinations of these instructions. A series of instructions is named a program. In order to process a program, the computer will have to load the program into its internal memory. The microcontroller fetches the instruction from the memory and processes the operation. Having completed the instruction the processor will fetch the next instruction and processes this one equally, repeating the process in a continues loop.

Flash-memory

In a standard PC you will have to 'start' a program, e.g. a game, before you are able to play the game. 'Starting' a program implies copying the program from your hard disc into the processor's memory. Switching off your PC will remove the program from the processor's memory and restarting the PC requires a 'restart' for the game-program to play the game.

A microcontroller however is provided with a special memory device, called 'Flash'-memory, in which the program is stored permanently. Switching off the supply power for the microcontroller will not remove the program from the 'Flash'-memory and to remove or modify a program, you will even have to switch on your microcontroller's power supply.

In fact a microcontroller uses 'Flash'-memory as a permanent processor memory, storing the program at any time until you decide to delete the program.

Loading a program

How do we load a program into the microcontroller's 'Flash'-memory?

Normally you would need a special programming device, equipped with an IC-socket, in which you would insert the microcontroller-chip. The programming device allows you to transfer the program directly into the microcontroller's 'Flash'-memory. The programming device is named 'programmer'.

Using the 'programmer' normally implies providing the YETI with some connector and cabling system and connecting the YETI with the 'programmer' to load a new program or program version. However a few clever engineers at DLR (www.dlr.de) were able to develop and provide us with a smart programming tool.

This smart tool does provide a permanent small piece of software in the microcontroller's 'Flash'-memory, called the 'Bootloader'. Immediately after switching on the microcontroller's power supply the microcontroller will always check whether a 'Bootloader'-program is available in the 'Flash'-memory. If the program is available, the microcontroller will execute this 'Bootloader'-program.

Now pay attention:

The 'Bootloader'-program has been designed to communicate with the YETI by infrared transmitter and receiver modules. Communications require special software at a PC. This software is called 'Flash' and may remind you of the name of the microcontroller's memory. Of course the PC will have to be equipped with equivalent infrared transmitter and receiver modules.

The bootloader and the 'Flash'-program will start communicating and the 'Flash'-program will now start sending a YETI-program to the 'Bootloader' instruction by instruction. The bootloader will receive the instructions one by one and will put them into its 'Flash'-memory. Of course the storing process will not overwrite the memory area, in which the 'Bootloader' is stored. Instead the received program will be stored in the memory area next to the 'Bootloader'-program.

After the transmission of a program the microcontroller stops executing the bootloader and starts the execution of the received and newly stored program.

The bootloader occupies a permanent and protected area in the normal microcontroller 'Flash'-memory and cannot be altered or deleted without special tools. The area is protected against erasure and modifications and you will feel safe at programming the system.

Developing programs

How do I start developing a program?

In fact we will use the following method:

You will 'write' a YETI-program in a text editor, using a special 'language'. Having completed the text file you will translate the text file to another file containing instructions, which may be understood by the microcontroller system and is called a 'hex'-file. And finally we will use the previously described 'Flash'-program to transfer the 'hex'-file to the YETI.

We might even consider writing a program for the microcontroller by editing 'hex'-instructions! Although you may successfully complete small programs, you will soon consider this method being too complicated and return to the standard programming method.

Writing programs

Programmers are used to name the developing process 'Writing programs', because a program basically must be considered to be plain text and sometimes may be compared to a simple letter or a short story. For program writing we will use a special text editor, called 'Programmers Notepad2' or PN2 in short. To write a YETI-program you might even use any other editor, such as Notepad or Microsoft Word, but we strongly advise to use PN2. Programmers Notepad2 is a special tool for writing computer programs, providing color-coding as a helpful aid in the programming process.

A program consists of a number of plain text lines, which of course must satisfy some prerequisites to be translatable into processor instructions. Another name for the plain text program is 'source code'. The prerequisites and conditions for translation into processor instructions make up the programming 'language'. And comparable to dictionaries in human languages, we use special translation programs to translate a plain text file, containing our 'source code', into a 'hex'-file with instructions for the microcontroller.

We can choose several languages for programming, but the most popular language for writing microprocessor software is a language with a short name: 'C'. This is the language we choose to write the YETI's software.

Compiling a program

A microcontroller is unable to understand the 'C'-language in the source code file and we will need a translator program to generate the instructions for the microcontroller. The translator program is just an ordinary program, translating one file into another. In information technology this kind of program is named 'Compiler'. So we will need a 'C'-compiler to create YETI-programs and consequently we choose the program 'gcc.exe'. The compiler will need an input file with a name extension dot and a letter 'c'. For this reason the compiler will be unable to translate a source file named 'test.txt', but it will process a file 'test.c' with the same contents.

Having customized the Programmers Notepad2 correctly you may press a button in the Tools->Make menu and wait patiently for the 'C'-compiler completing the translation of your source file into a 'hex'-file.

'Uploading' a program

Imagine the compiler has just completed the translation of a 'C' program 'test.c' into a file with microcontroller instructions called 'test.hex'. As a final step you may start the flash program to transfer the 'hex'-file by infrared communication to the YETI microcontroller. This final step, in which the flash program transfers the 'hex'-file to the YETI, is called 'Uploading' a program.

Customizing a compiler system

If you plan to buy a 'C'-compiler, you will normally have to invest a few hundred Euros, Dollars or Pounds. However a group of professional programmers invested a lot of time and effort to develop an extremely versatile 'C'-compiler system, which is even free of charge!

The WINAVR programmers prepared the compiler system and the Programmers Notepad2 for a standard default configuration. The user however may vary the default configuration ad lib.

The default configuration requires the input source file has to be called 'test.c'. Parallel to the source file we will need a file 'YETI.c' optionally containing additional functions and allowing separating the source codes for the main program and additional function coding. As a programmer all you need to do is press a button and the Programmers Notepad2 will take care of compiling all required source files.

You may think: 'Seen it and been there'. Yes, you may have read these lines before. But it may be much clearer now than before.

Writing a YETI 'C' -program

Is it difficult to write a YETI 'C' -program?

Well the answer may be negative or positive as well. It is rather easy, because a YETI-control program is rather simple. However we can also create quite complex programs for the YETI.

If you start working with 'C' things may seem to be magic, but soon you will learn the magic is quite simple. You don't have to understand all individual steps in order to use them. Keep on reading and in the end you will be seeing things clearly.

Basic structure for a 'C'-program

Basically a 'C'-program minimally requires the following structure:

```
int main(void){  
    return 0;  
}
```

'int'	is the type for the main function
'main'	is the name for the main function
'void'	indicating 'no entry'
'return 0'	is the return value for the 'main'-function.

As a general rule each 'C'-instruction has to be terminated by a semi-colon, except a block terminator (terminating bracket).

Programmers Notepad2 will automatically display the source code in predefined colors according to some special categories. Syntax or specific 'C'-keywords will be colored green, whereas numbers are displayed in red and comments will be written in blue.

You may insert comments virtually anywhere, following '/' or between '/*' and '*/'

```
int main/*intermediate text line*/(void){ /*any text lines*/  
    //one line of text  
    /* text lines  
    */  
    return 0; //any text  
/*  
some more text lines  
*/  
}
```


The following program source will generate a compiler error message, if we write 'Main' with a capital letter. The 'C'-language does not allow us to write the keyword 'main' to be written with a capital letter.

```
int Main(void){  
    return 0;  
}
```

On the other hand the compiler allows us to concatenate all instructions in one single line, but this will not improve the readability:
int main(void){return 0;}

#Comments can be appended at the end of line:

```
int main(void){                //main function entry  
    return 0;                  //terminating the main function and  
                                returning a value 0  
}                               //end of the main function
```

#Comments can also be inserted above the program lines:

```
// main function entry  
int main(void){  
// terminating the main function and returning a value 0  
    return 0;  
// end of the main function  
}
```

‘C’-language always requires one single main function named ‘main’. Additional functions may be inserted ad lib.

The following program, containing merely one main function ‘main’, will not be processing any instruction and may be seen as a simple frame:

```
int main(void){                // main function entry
    return 0;                  // terminating the main function and returning a value 0
}                               // end of the main function
```

‘C’-Course

As already stated, we would be glad to present a complete ‘C’-course in this manual. However a lot of excellent books and a great number of websites explain basics and details in the ‘C’-language already. For this reason we will restrict our descriptions to functions, which will be needed for the YETI.

Having understood the basics we can start programming now. We will proceed by explaining the sample programs included in the kit’s CD. Our explanations will be a fine practical training course for beginners and a retraining course for others.

IMPORTANT HINT

The letter **v** at the beginning of YETI function names is a valuable programmer's hint, indicating a function, which will not be returning a value!

YETI will switch on its right 'eye'

```
#This program will switch on YETI's right 'eye'
#include "YETI.h"           //load definitions and functions
int main(void){             //main function entry
    vInitYeti();             //initialize all microprocessor modules in
                             //the YETI
    vFrontLEDs(RIGHT);      //switch on YETI's right 'eye'-LED
    return 0;               //terminating the main function and
                             //returning a value 0
}                             //end of the main function
```

#include "YETI.h"

At this location the compiler will insert file 'YETI.h', containing all definitions for functions, e.g. function 'vYetiInit()'. Coding lines for functions will be found in file 'YETI.c'.

vInitYeti();

This is a special function, initiating all microcontroller and YETI modules. The function may be called by a statement 'vInitYeti();' at the very beginning in YETI programs and its coding is to be found in file 'YETI.c'.

vFrontLEDs(LEFT);

This function will activate YETI's left eye LED.

Equivalent functions are:

vFrontLEDs(RIGHT);

Activating YETI's right eye.

vFrontLEDs(OFF);

Switches off both eyes.

vFrontLEDs(BOTH);

Switches on both eyes

Blinking YETI's right eye LED 5 times

#This program will activate YETI's right eye LED 5 times at one second intervals

```
#include "YETI.h"                                //load definitions and
                                                    //functions

int main(void){                                  //main function entry
    int i;                                       //define variable 'i' as an integer
    vInitYeti();                               //initialize all microprocessor
                                                    //modules in the YETI
    for(i=0 ;i<5 ;i++){                       //repeat for-loop (5 times)
        vFrontLEDs(RIGHT);                    //activate YETI's right eye LED
        vWaitMilliseconds(500);               //wait half a second
        vFrontLEDs(OFF);                      //deactivate YETI's right eye LED
        vWaitMilliseconds(500);               //wait half a second
    }                                           //terminate for-loop (5 times)
    return 0;                                  //terminating the main function
}                                              //end of the main function
```

```
for(...){  
}
```

The program module enclosed in 'for'-brackets '{' respectively '}' will be called a 'loop'.

int i;

This statement defines a Variable with a free chosen name 'i'.

```
for(i=0 ;i<5 ;i++){
```

Initiate variable 'i' with value 0 and repeat all program coding between the opening bracket '{' and the corresponding closing bracket '}', as long as variable 'i' is less than 5. Any time the program meets the 'for'-line in the source coding the value in 'i' is incremented by 1. The very first time the program meets the 'for'-line in the source coding the value in 'i' is 0.

At the 6e passage 'i' reaches the value 5 and no longer meets the condition $i < 5$, which will force the program to continue at the next coding line, following the closing bracket '}' at the 'for'-loop. In our example this line reads: return 0;

The program executed the 'for'-loop 5 times, which is exactly what we wanted to be done.

vWaitMilliseconds(500);

Calling function vWaitMilliSeconds and insert a variable 500.

This function can be found in file 'YETI.c', which can be opened for reading in Programmers Notepad2. The function will be using an internal microcontroller counter, to wait for a prescribed number of milliseconds and then return to the calling program. The function will accept a number of milliseconds (1/1000th second) and the above calling sequence will generate a waiting cycle of half a second.

YETI's acoustic signals

#The following program will generate a sample of YETI's acoustic signals.

```
#include "YETI.h"           //load definitions and functions
int main(void){             //main function entry
    vInitYeti();             //initialize all microprocessor
                              //modules in the YETI
    vBeep(400,80);           //make YETI generate a beep sound
    vBeep(520,120);          //make YETI generate a beep sound
    vBeep(360,80);           //make YETI generate a beep sound
    vBeep(580,160);          //make YETI generate a beep sound
    return 0;                //terminating the main function
}
```

vBeep(400,80);

will call function 'vBeep' at a pitch of 400 and a duration of 8 x 10 = 80 milliseconds.

Move the servomotor for YETI's legs and feet.

#This program will make YETI jump for joy.

```
#include "YETI.h"           //load definitions and functions
#include "yetimove.c"        //insert YETI's servo functions
int main(void){             //main function entry
    int i;                  //define variable 'i' as an integer
    vInitYeti();            //initialize all microprocessor modules in
                            //the YETI
    for(i=0 ;i<3 ;i++){     //do for-loop 3 times
        vMoveBody(16,10);   //move body to the left
        vMoveBody(-16,10);  //move body to the right
    }                       //terminate for-loop (3 times)
    vMoveBody(0,10);        //move body to the centre position
    return 0;               //terminating the main function
}
```

#include "yetimove.c"

This include-line will insert the YETI servo functions.

vMoveBody(16,10);

Move YETI's feet servomotor 16 steps to the left, reserving 10 milliseconds for each step.

vMoveBody(-16,10);

Move YETI's feet servomotor 16 steps to the right, reserving 10 milliseconds for each step.

You may choose step positions between -58 and +58. The delay time period will always be rounded up to the next higher multiple value of milliseconds, resulting in a delay of 10 milliseconds/step for each parameter value between 1-10 milliseconds and a delay of 20 milliseconds/step for each parameter value between 11-20 milliseconds

vMoveBody(0,10);

Reset the feet-servo for the YETI body to an initial position, resulting in an upright body position for the robot.

A similar function 'vMoveLegs()' will activate YETI's legs forward and/or backward.

YETI starts walking

#This program will activate YETI to walk 3 steps forward.

```
#include "YETI.h"           //load definitions and functions
#include "yetimove.c"        //insert YETI's servo functions
int main(void){              //main function entry
    vInitYeti();              //initialize all microprocessor modules in
                              //the YETI
    vMoveForwardXSteps(3);    //YETI will march forward 3 steps
    vStandUpright();          //makes YETI stand upright
    return 0;                 //terminating the main function
}
```

vMoveForwardXSteps(3);

YETI will march forward 3 steps. Whenever YETI starts from an upright position, it will always start putting its right foot forward.

vStandUpright();

Makes YETI stand upright

14. YETI EXTENSION SETS

General overview

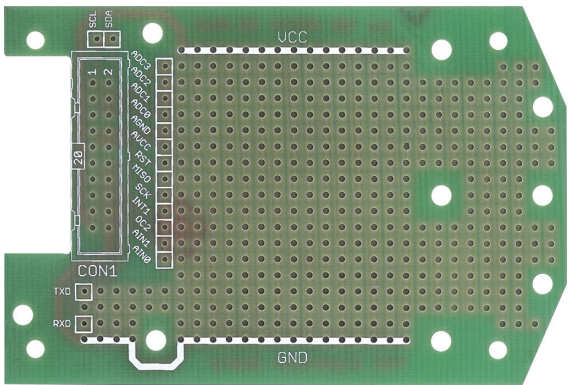
All extension sets will be connected to YETI's main PCB using a single 20-pole flat cable. The flat cable will also provide supply power to the sets and the I2C datatransfer to and from the extension sets.

14.1. YETI Experimental set YT-EXP1

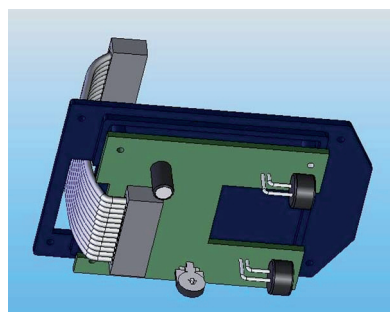
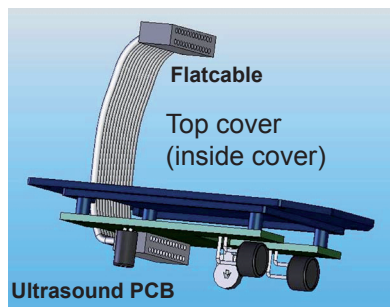
The experimental extension set has been designed to set up your own electronic designs and connect the experimental modules to the microcontroller.

14.1.2. Partlist Experiment KIT YT-EXP1

PCB-DSP		YETI Experiment PCB
CON1-PCB		PCB connector, male, 20 pins, for flatcable
CON1-FC	(2 pcs.)	Flatcable connector, 20-pins
F1		Flatcable, 20-ires, 10cm



INSTALLING THE UPGRADE KITS



Installing Ultrasound PCB



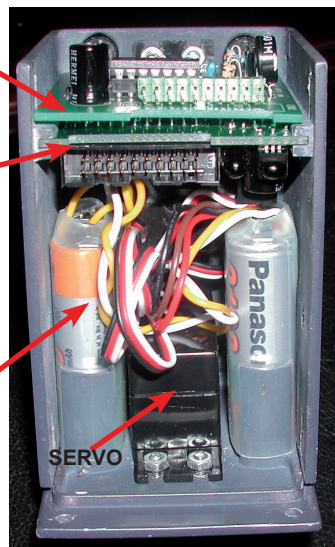
Installing experiment PCB

Ultrasound PCB

Main PCB

Accu Set

SERVO



Installing
Display PCB



14.2. YETI Display kit YT-DSP2

General overview

The additional display module provides four 8-segment symbol displays. The display module also contains a 24-pin I2C driver chip for symbol display control.

The YETI microcontroller controls the I2C driver chip. I2C is a standard communication protocol between electronic components, using only two signal wires SCL (serial clock) and SDA (serial data). We will need just one pair of a total of 20 wires provided by the flat cable.

The digits allow us to display a number of messages or values. The simple basic principle of the I2C chip allows us to create and display a number of new symbols.

Hardware description

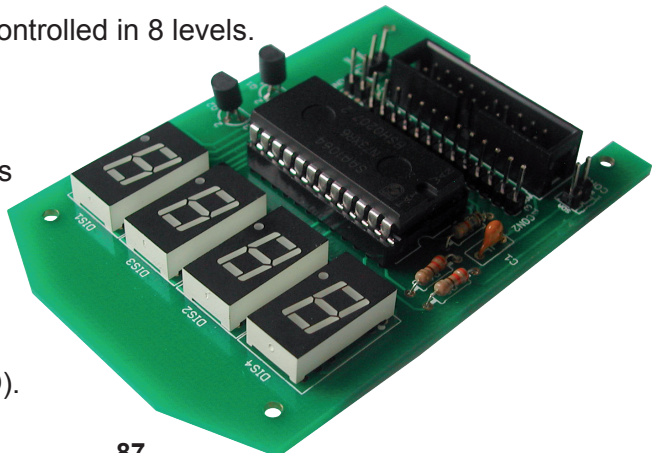
The display system provides four 8-segment displays, an I2C display chip and a few supplementary components.

Using the I2C-signals the chip communicates with a control unit, which will normally be a microcontroller. I2C is a standard communication protocol between electronic components, using only two signal wires. The chip will take care of all control signals for the displays and the user merely needs to define which symbol needs to be displayed at each of the display positions. Control signals of course will use the I2C-bus system.

Display intensity can be controlled in 8 levels.

Each display provides 8 signal lines for control, 1 control line for each segment. Seven data-lines control the 7 segments and one line controls the decimal dot.

The 7 segments and the dot each contain a Light Emitting Diode (LED).



Additionally each display provides a common supply pin for the LED's. Four segments each share a common supply pin and both supply pins 3 and 14 are interconnected internally.

In order to provide control signals for 4 display units we normally would need at least $8+2=10$ signal lines for each symbol, requiring a 40-pin IC. However we can use a tricky multiplex system, providing 2 sets of 8-segment pins: P1-P8 and P9-P16. Let's first have a look at the first set P1-P8, which is connected to display 1 and display 2 simultaneously. Feeding P1-P8 with a certain bit combination for a special display symbol, e.g. "X", the units 1 and 2 would both display the same symbol "X". Now we just need to activate display 1 and to deactivate display 2 with switching transistor Q1 and Q2, merely activating display 1.

In a next step the chip will switch off display 1 by deactivating transistor Q2, provide a new bit combination for a new display symbol, e.g. "Y", feeding the combination to P1-P8, and switch on transistor Q1 to activate display 2. The same procedure will be used for display 3 and 4. Using a high switching rate, which is invisible to the human eye, we will not be able to observe the 50% dark phases, in which the symbols are switched off. The human eye will see the display symbols constantly at a reduced intensity.

An alternative display method allows using two displays without multiplexing. These displays can be activated without switching on and off. To do so one display (Display 1) must be using P1-P8, whereas the other display (Display 3) will be using P9-P16. The software example suggests the display units will be arranged in a special way:

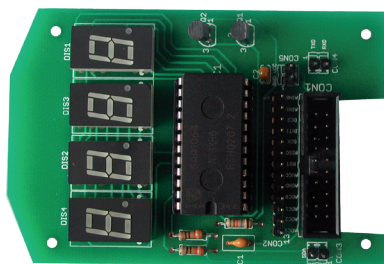
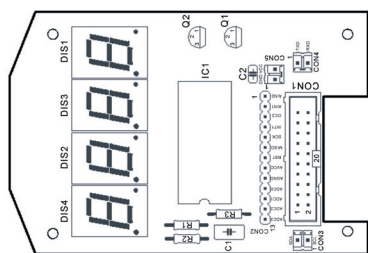
Display arrangement:

Display 4 Display 2 Display 3 Display 1

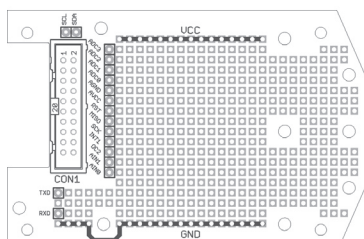
The basic idea of the arrangement for using two display units (display units 1 and 3) in a static mode is the requirement these elements must be neighbours.

14.4. Parts list YETI Display Kit YT-DSP2

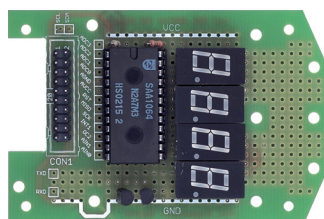
PCB-DSP	YETI DISPLAY PCB
R1	330R (<i>Or, or, brn, gld</i>)
R2	330R (<i>Or, or, brn, gld</i>)
R3	18K (<i>Brn, grey, or, gld</i>)
C1	2,7nF (<i>272</i>)
C2	100nF (<i>104</i>)
Q1	BC547B/C (<i>polarity!</i>)
Q2	BC547B/C (<i>polarity!</i>)
D1	8-segment display common anode (<i>polarity!</i>)
D2	8-segment display common anode (<i>polarity!</i>)
D3	8-segment display common anode (<i>polarity!</i>)
D4	8-segment display common anode (<i>polarity!</i>)
IC1	SAA1064 (<i>polarity!</i>)
S1	IC-socket, 24-pins, 600mil (<i>polarity!</i>)
CON2	Pin header 13 -pins, PCB montage
CON, 3, 4 and 5 (3. pcs.)	Pin header 2 -pins, PCB montage
CON1-PCB	PCB connector, male, 20 pins, for flat cable
CON1-FC	Flat cable connector, 20-pins, female
F1	Flat cable, 20-wires, 10cm

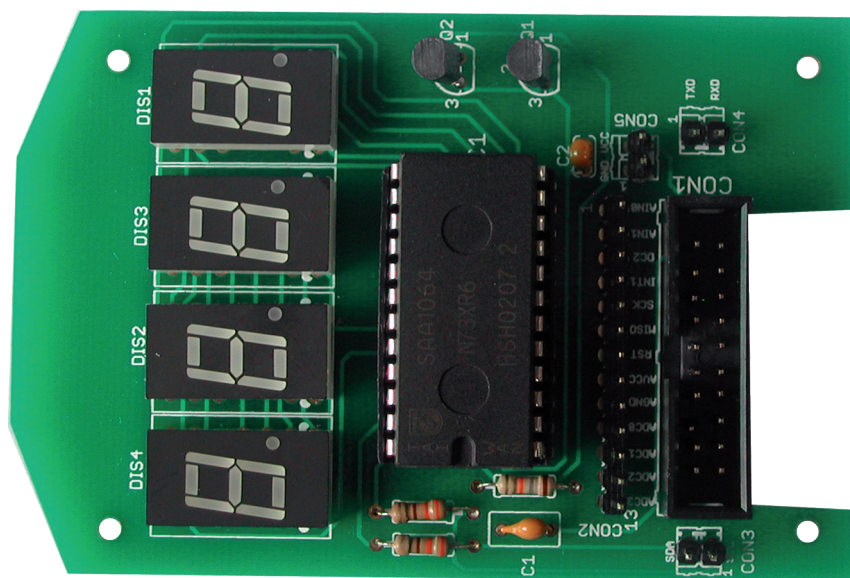
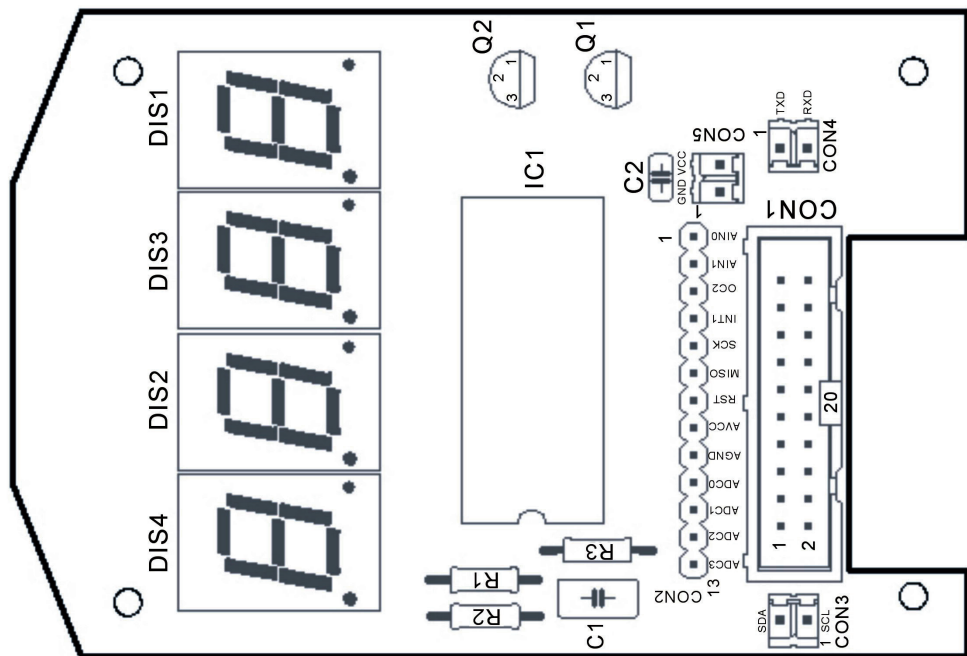


Display build on original Display PCB YT-DSP2 KIT

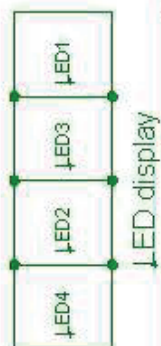
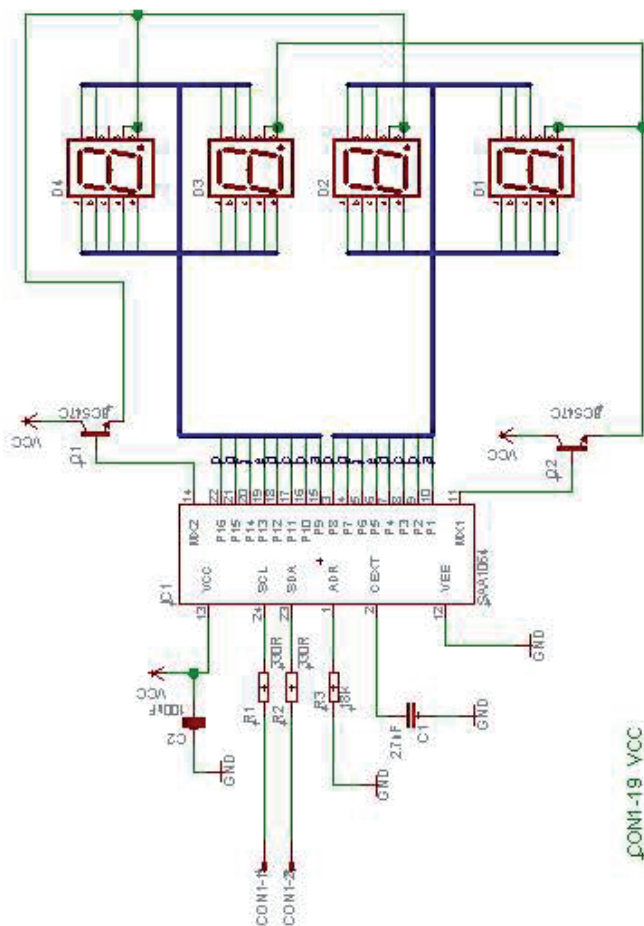


Display build on
Experiment PCB





14.5. Diagram Display set



AREXX

YETI display
2006-03-16 14:16:30
Sheet 1/1

14.6. YETI Ultrasonic kit YT-ULT3

General Overview

The ultrasonic extension kit contains ultrasonic transmitter and receiver modules. Ultrasonic waves are soundwaves at relatively high frequencies, which cannot be heard by human ears.

Hunting bats however use ultrasonic waves for orientation while flying in completely dark areas, avoiding all obstacles and catching their preys. We call these ranging methods echoranging. Obstacles and preys reflect the soundwaves and the bat's ears detect the reflected waves.

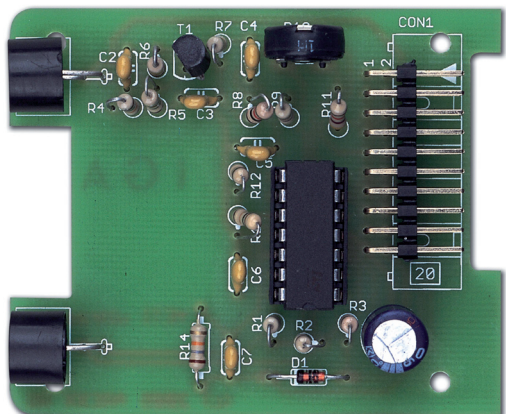
YETI instead uses a microphone for sound detection.

The transmitter will emit acoustic impulses (wave bursts) with frequencies around 40.000Hz. The receiver will detect signals, reflected by nearby objects, and also the delay between transmission and reception events.

The delay between transmitted and received impulses allows us to calculate the distance between transmitter/receiver and reflection area. The ultrasonic module converts the delay period into an electronic voltage level and a flat cable connects the delay signal to an A/D-converter in the YETI processor, monitoring the voltage level.

A software module controls the voltage monitoring and all resulting robotic actions. We did reserve sufficient room in YETI's head to install the ultrasonic modules.

The transmitter and receiver each will be hidden behind an "eyebrow"-hole in the forehead.



14.7. Hardware description

The ultrasonic module consists of 5 parts:

1. Transmitter
2. Receiver
3. Receiver amplifier
4. Fixed voltage reference
5. Variable voltage reference

The microcontroller generates the ultrasonic signal wave to be transmitted by the transmitter loudspeaker (TX). The receiver microphone (RX) receives reflected sound waves, which must be amplified in the receiver amplifier. Resistor R10 allows you to manually control the amplification factor. The fixed voltage reference, which is exactly adjusted to 50 % of the supply voltage, will be used for the transmitter and for generating the variable voltage reference.

The variable voltage reference controls the hearing sensitivity. On every transmission impulse the microcontroller will adjust its sensitivity, increasing sensitivity with delay and distance. A growing distance will result in weaker reflections signals and in growing delays as well.

The microcontroller generates the ultrasonic signal wave, entering the ultrasonic module at pin CON1-13. The reflected signal, as monitored in the microphone, is returned to the microcontroller by pin CON1-6. At any transmitted impulse, pin CON1-15 will ramp down a voltage signal for the microcontroller validation.

Of course the microcontroller will not generate an acoustic but an electronic signal. In fact the loudspeaker will generate the ultrasonic sound waves from the electronic signal.

The generated 'ultrasonic' signal will leave the microcontroller and enter the transmitter module by CON1-13 and resistor R3. The transmitter consists of 2 individual amplifiers in a chip IC1 (IC = Integrated Circuit), containing a total of 4 amplifiers. In electronics these amplifiers are named Opamp (Operational Amplifier), containing a differential amplifier stage using two input ports: a positive and a negative entry port. The differential amplifier stage will process the voltage difference between both ports and the opamp's output voltage will be proportional to the voltage difference between both ports.

The ultrasonic signal will now be applied to one entry port of two opamps. The other entry ports of both opamps are supplied with a fixed voltage reference of exactly 50% of the supply voltage. Both opamps are needed to generate a maximal energy for the loudspeaker and to regenerate the weak and distorted microphone signal. The ultrasonic microphone (RX) detects the reflected signal and transforms it into an electronic signal, which may be filtered and amplified in receiver amplifier Opamp IC1B. An adjustable resistor allows you to control the amplification factor of the system.

As resistor R3 leads the ultrasonic output pulses to the loudspeaker, the signal will also pass diode D1 and load capacitor C7 immediately to a full supply voltage value VCC, leading to a sharp raise of a voltage at pin CON1-15 at the beginning of transmission impulses. At the trailing edge of the transmission impulse capacitor resistor R14 will discharge C7.

The microcontroller contains an analogue comparator in order to compare two voltage levels: the received signal level at pin CON1-6 and the decreasing voltage level at pin CON1-15.

If the received signal level exceeds the decreasing voltage level at pin CON1-15, the microcontroller will accept the signal as a valid reflection signal. Using the decreasing comparison level will result in high validation signal levels for fast response reflections and gradually lower validation signal levels for retarded response reflections.

Of course the ultrasonic receiver system is extremely sensitive for any reflected signals, especially for signals from nearby objects in the transmitter's vicinity.

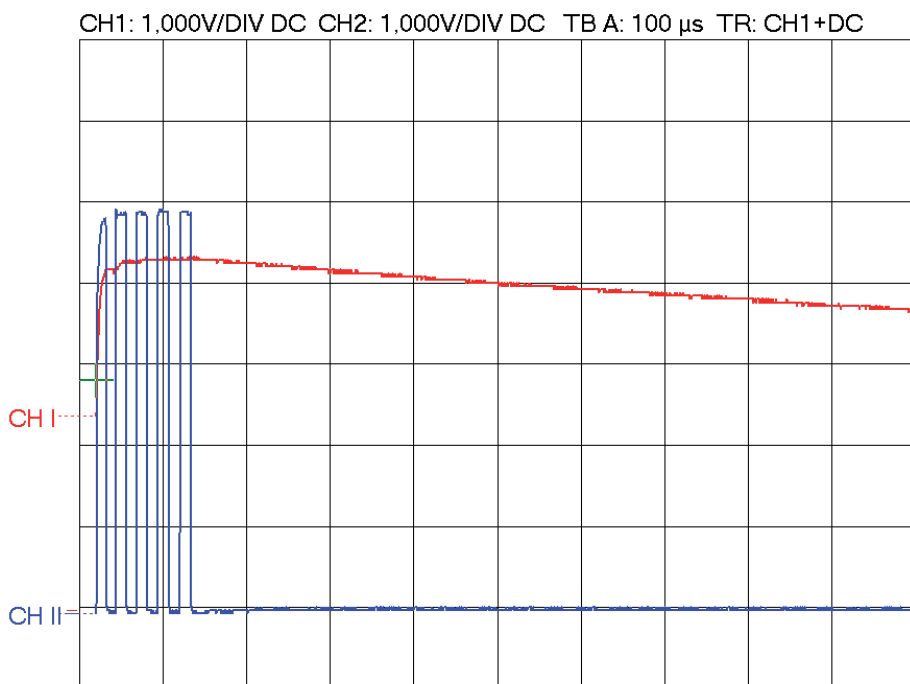
In order to prevent reflections from the YETI's head surrounding the ultrasonic transmitter and receiver, we will completely fill the robot's head with cotton wool, including the volume between the band cable and the back of the head. See figure 1.

14.8. Preparation YETI Ultrasonic Set

The assembled ultrasonic PCB is mounted in the YETI head. However, in this location the functioning of this ultrasonic set will be influenced by:

1. Undesired reflections of the ultrasonic sound inside the YETI head.
2. Undesired reflections of the ultrasonic sound by the outside of YETI itself.

Below you find two important ultrasonic signals:

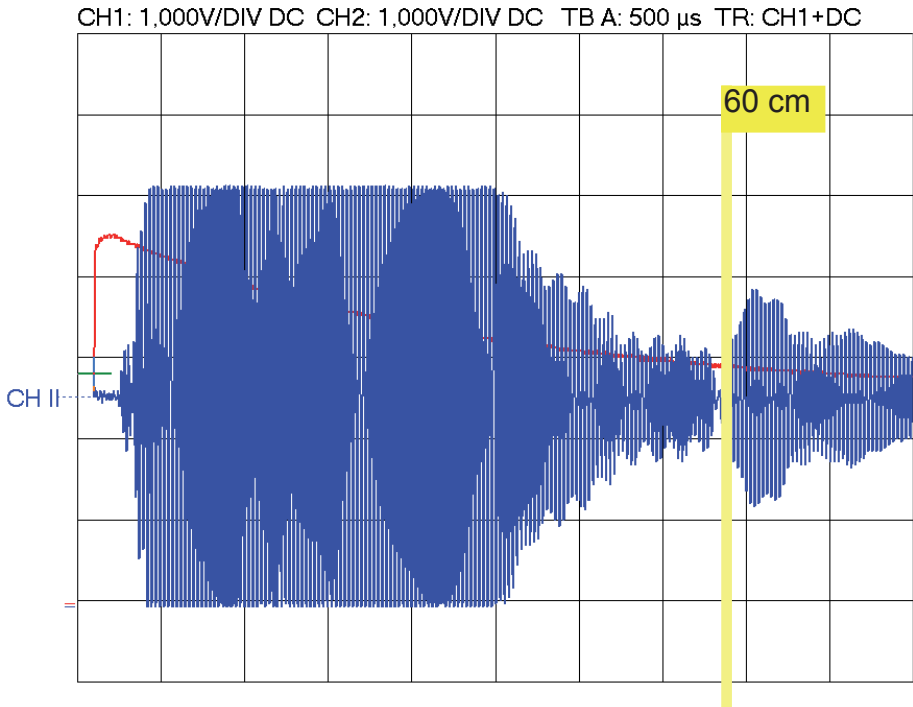


On this oscilloscope image you see two signals, measured on CON1-13 and CON1-15.

The first signal blue (CON1-13) briefly shows 5 pulses of approx. 4.5 Volt. The signal comes from the microprocessor directly and goes to the ultrasonic PCB.

The second signal red (CON1-15), an unloading curve, is a reference signal coming from the ultrasonic PCB. It returns to the processor.

If a reflection finds itself above the reference signal, it will be seen as a valid measurement.



On the above image you see two signals:

- ***the already mentioned red reference signal CON1-13 and***
- ***the receiving blue signal CON1-6 (the signals received and reflected by the ultrasonic PCB, then going to the microprocessor through CON1-6).***

We mark the reflected signal at a distance of 60 cm. All strongly reflected signals are located above the reference line as a result of undesired direct reflections of YETI. These signals therefore cause an invalid measurement.

As you can see in our description the US receiver reacts very sensitive on all kind of reflected signals. This means it also reacts on false reflections inside YETI's head.

In order to avoid invalid measurements, we put cotton wool inside the YETI head. This way the inside of the YETI head does no longer cause undesired reflections. (see. fig.1)!

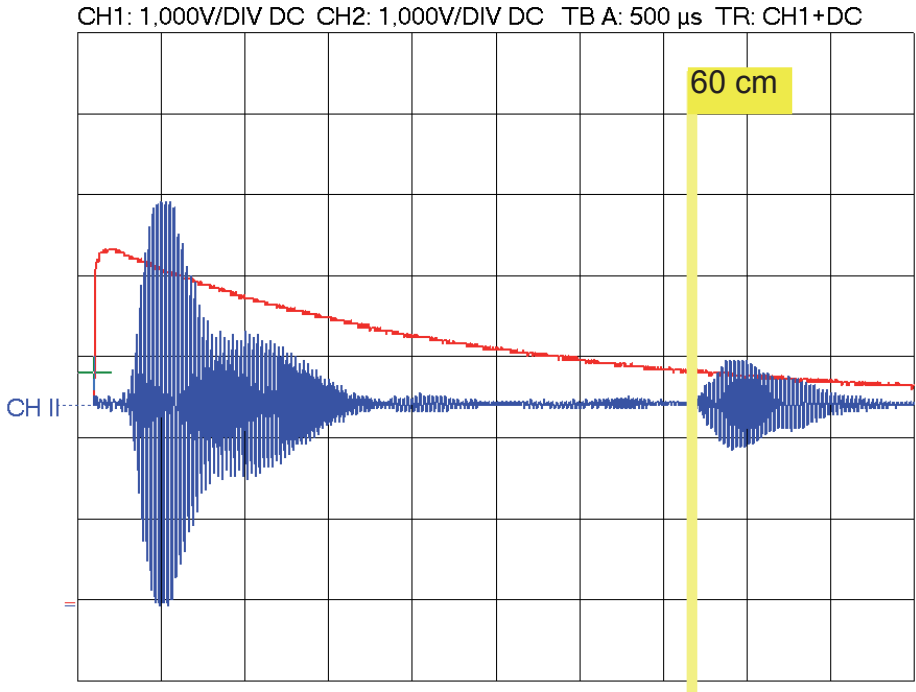


Step 1.

Yeti's head, completely filled with cotton wool.

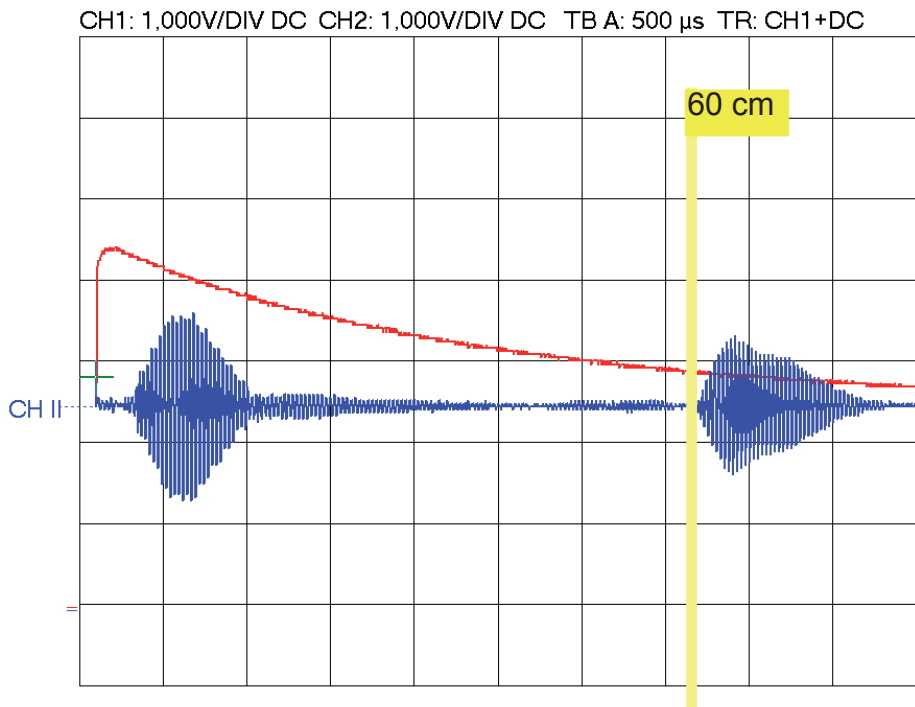
The image below shows this new situation:

It shows perfectly that there are clearly less undesired reflections now. Reflections which are stronger than the reference signal cannot be seen as valid measurements, they are error measurements.



Step 2.

Filtering the outside reflections



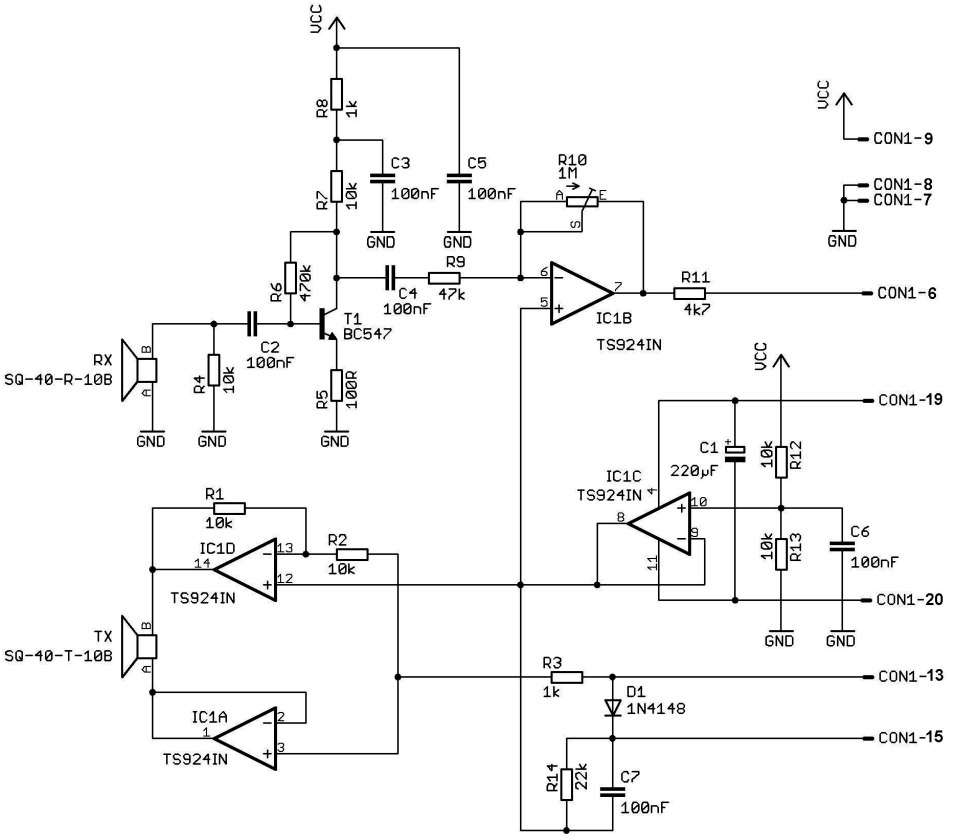
On the above image all reflected signals stay under the reference line. Now you can measure distances accurately. The time between 5 transmitted pulses and the reflected signal is calculated by the microprocessor and translated into an actual distance. This distance can be shown on a display.

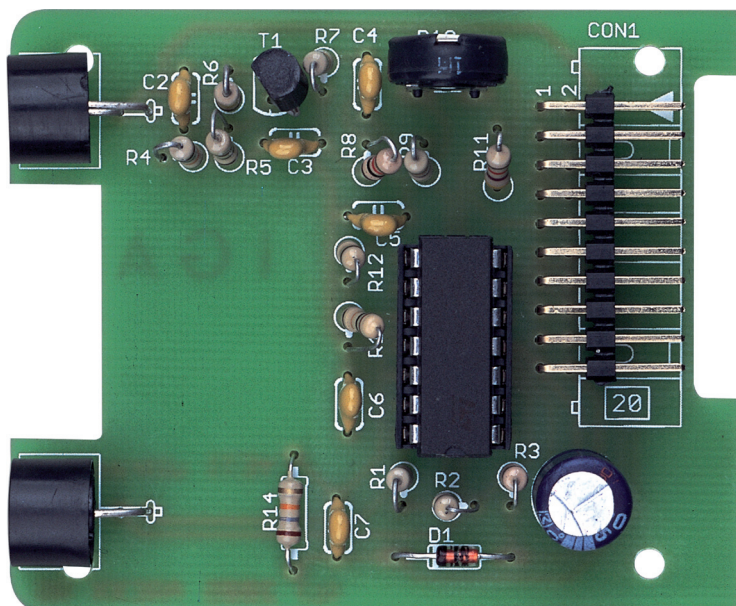
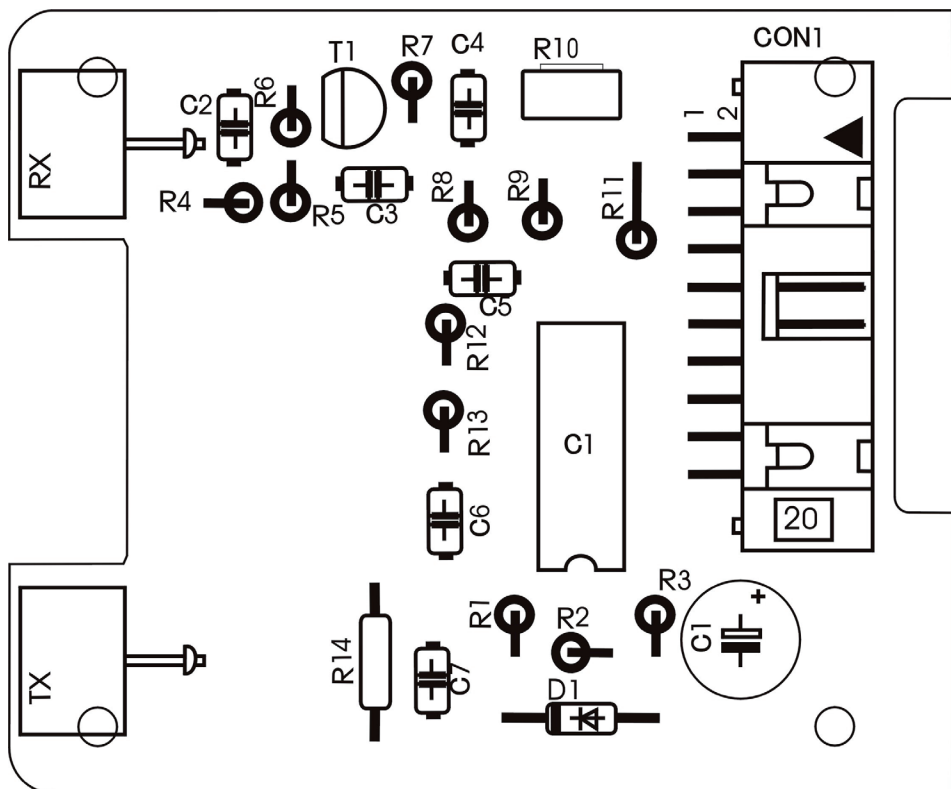
It is very important that during the preparation of the ultrasonic function you take care that undesired reflections stay under the reference line.

14.9. Parts list YETI Ultrasonic set

PCB-UTS	Ultrasonic PCB
R1	10K (<i>Brn, blk, or, gld</i>)
R2	10K (<i>Brn, blk, or, gld</i>)
R3	1K (<i>Brn, blk, red, gld</i>)
R4	10K (<i>Brn, blk, or, gld</i>)
R5	100R (<i>Brn, blk, brn, gld</i>)
R6	470K (<i>Ylw, vio, ylw, gld</i>)
R7	10K (<i>Brn, blk, or, gld</i>)
R8	1K (<i>Brn, blk, red, gld</i>)
R9	47K (<i>Ylw, vio, or, gld</i>)
R10 TRIMMER	1M potentiometer
R11	4K7 (<i>Ylw, vio, red, gld</i>)
R12	10K (<i>Brn, blk, or, gld</i>)
R13	10K (<i>Brn, blk, or, gld</i>)
R14	22K (<i>Red, red, or, gld</i>)
C1	220uF (<i>polarity!</i>)
C2	100nF (<i>104</i>)
C3	100nF (<i>104</i>)
C4	100nF (<i>104</i>)
C5	100nF (<i>104</i>)
C6	100nF (<i>104</i>)
C7	100nF (<i>104</i>)
IC1	TS924IN (Quad Opamp) (<i>polarity!</i>)
S1	IC-socket, 14-pins (<i>polarity!</i>)
TX	400ST100 (Ultrasonic transmitter) small
RX	400SR100 (Ultrasonic receiver) small
T1	BC547B/C or BC547B/C (<i>polarity!</i>)
D1	1N4148 (<i>polarity!</i>)
CON1-PCB	PCB connector, male, 20 pins, for flat cable
CON1-FC (2 pcs.)	Flat cable connector, 20-pins, female
F1	Flat cable, 20-wires, 10cm

14.10. Diagram Ultrasonic set





Conclusion

We hope our robots ASURO and YETI may have helped you by introducing you into the world of robotics. We believe the next technological revolution will be a robotics revolution. Robots will contribute to economic growth as well and in order to support growth we must include robotics in technological education,

resulting in a mission statement for the development team of ASURO and YETI:

TO TRAIN A SCIENTIFIC MIND



APPENDIX

A. OVERVIEW OF YETI FUNCTIONS

Basic functions.

The following overview describes YETI's basic functions, to be found in file 'YETI.c'.

vInitYeti()

Initialize all YETI modules.

vFrontLEDs(x)

x = ON, LEFT, RIGHT, OFF

Activate the YETI's eye-LED's.

Example:

```
vFrontLEDs(LEFT);
```

vServo1ToPosition(x)

x = 0-65635

Set the front (body) servomotor into position x.

WARNING: This function may set the servomotor into any position, including mechanical positions, which are unaccessible for the YETI system. These commands may damage the YETI and we suggest you to use the safer function 'vMoveBody()'.

Example:

```
vServo1ToPosition(35);
```

vServo2ToPosition(x)

x = 0-65635

Set the front (legs) servomotor into position x.

WARNING: This function may set the servomotor into any position, including mechanical positions, which are unaccessible for the YETI system. These commands may damage the YETI and we suggest you to use the safer function 'vMoveLegs()'.

Voorbeeld:

```
vServo2ToPosition(35);
```

vRs232Write(x,y)

x = text

y = text length, 0-255

Example:

```
void vRs232Write("Hello World",11) ;
```

vRs232Read(x,y,z)

x = pointer to an array to store a text for reception

y = expected text length

z = timeout

Example:

```
char RxData[10] ;
```

```
vRs232Read(&RxData[0],4,0);
```

Having received 4 symbols the function will be terminated.

vWaitMilliseconds(x)

x = waiting time in milliseconds, 0 - 65635

This function will round up duration periods in ten milliseconds, e.g. uprounding 23 to 30.

Example: wait 1 second.

```
vWaitMilliseconds(1000);
```

vBeep(x,y)

x = pitch number, 1-3906

y = duration in milliseconds.

Example:

```
vBeep(200,100);
```

Compound functions

The following functions are compound functions, containing an assembly of one or more basic functions. Compound functions will be found in the file 'yetimove.c'.

vStandUpright()

Will make YETI to stand up in an upright position. Both 'body'- and 'legs'-servosystems will be reset to a zero position.

vMoveBody(x,y)

x = YETI's body is inclining to the left or right position, from -58 (corresponding to an extreme right inclination) up to and including +58 (corresponding to an extreme left inclination)

y = execution speed in milliseconds / step, 0 – 65635.

This function will round up delay durations to tens of milliseconds, e.g. uprounding 17 to 20.

Example:**vMoveBody(-25,20);**

From an upright position the command vMoveBody(-25,20) will need $25 \times 20 = 500\text{ms} = 0,5$ seconds to incline YETI to the right side at a position of -25.

vMoveLegs(x,y)

x = YETI's right leg will be moved forward or backward (and the other leg will be moved in the opposite direction), between -58 (right foot forward) up to and including +58 (left foot forward)

execution speed in milliseconds / step, 0 – 65635.

This function will round up delay durations to tens of milliseconds, e.g. uprounding 3 to 10.

Example:**vMoveLegs(-25,20);**

From an upright position the command vMoveLegs(-25,20) will need $25 \times 20 = 500\text{ms} = 0,5$ seconds to move YETI's right leg forward to position -25 (while moving its left leg backward).

vMoveForwardXSteps(x)

x = number of steps forward, 0-255

Example:**vMoveForwardXSteps(3);**

YETI will walk 3 steps in forward direction. From an upright position YETI will always start by moving its right leg in a forward direction.

vMoveBackwardXSteps(x)

x = number of steps backward, 0-255

Example:**vMoveBackwardXSteps(4);**

YETI will walk 3 steps in backward direction. From an upright position YETI will always start by moving its right leg.

vTurnLeftXSteps(x,y)

x = number of steps, 0-255

y = 'true' respectively 'false', true = moving forward, false = moving forward.

Example:

vTurnLeftXSteps(2,false);

YETI will turn around backward, moving backward 2 steps in a left turn.

vTurnRightXSteps(x,y)

x = number of steps, 0-255

y = 'true' respectively 'false', true = moving forward, false = moving forward.

Example:**vTurnLeftXSteps(3,true);**

YETI will turn around forward, moving forward 3 steps in a right turn.

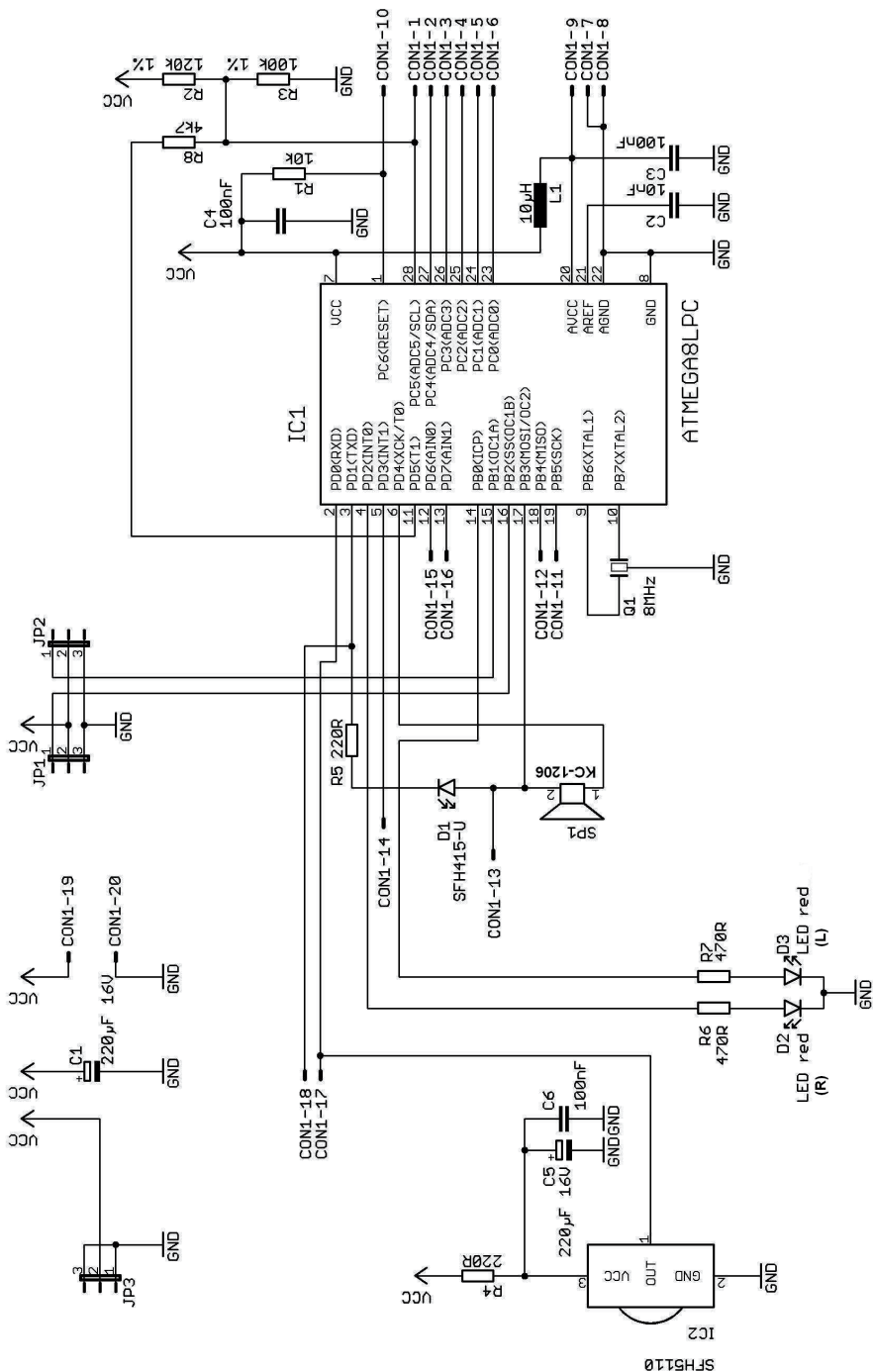
vCalibrateServos()

In a combination with the free communication software Hyperterminal this function will be needed to recalibrate the servo system each time you adjust the mechanical positions of servomotors or servomotor-limbs. The function may also be used for remote controlling the YETI by Hyperterminal keyboard, using keys 'W', 'A', 'S' and 'D'.

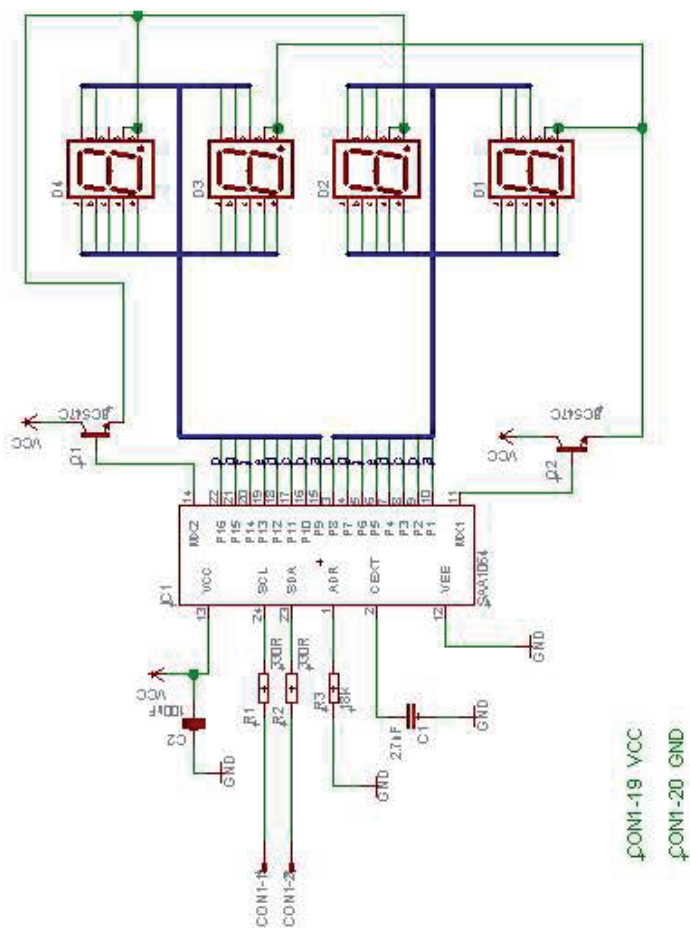
Hint:

Having calibrated the YETI's servos, you may remove this function from the file, resulting in a considerable reduction of the program's file size.

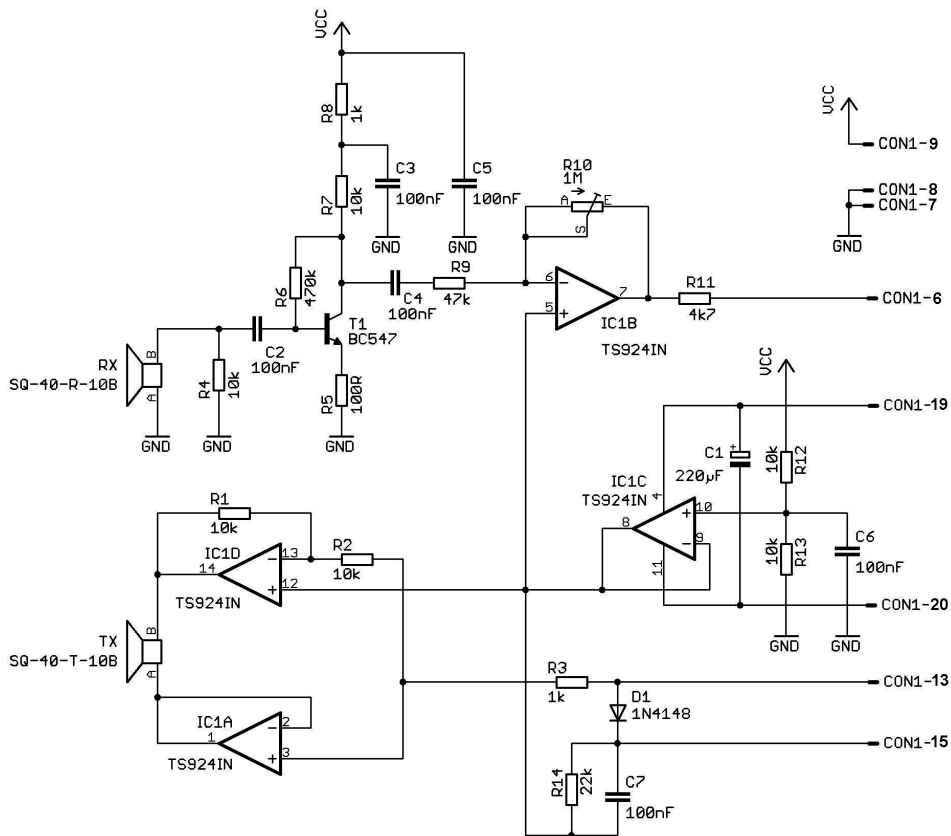
B. DIAGRAM YETI



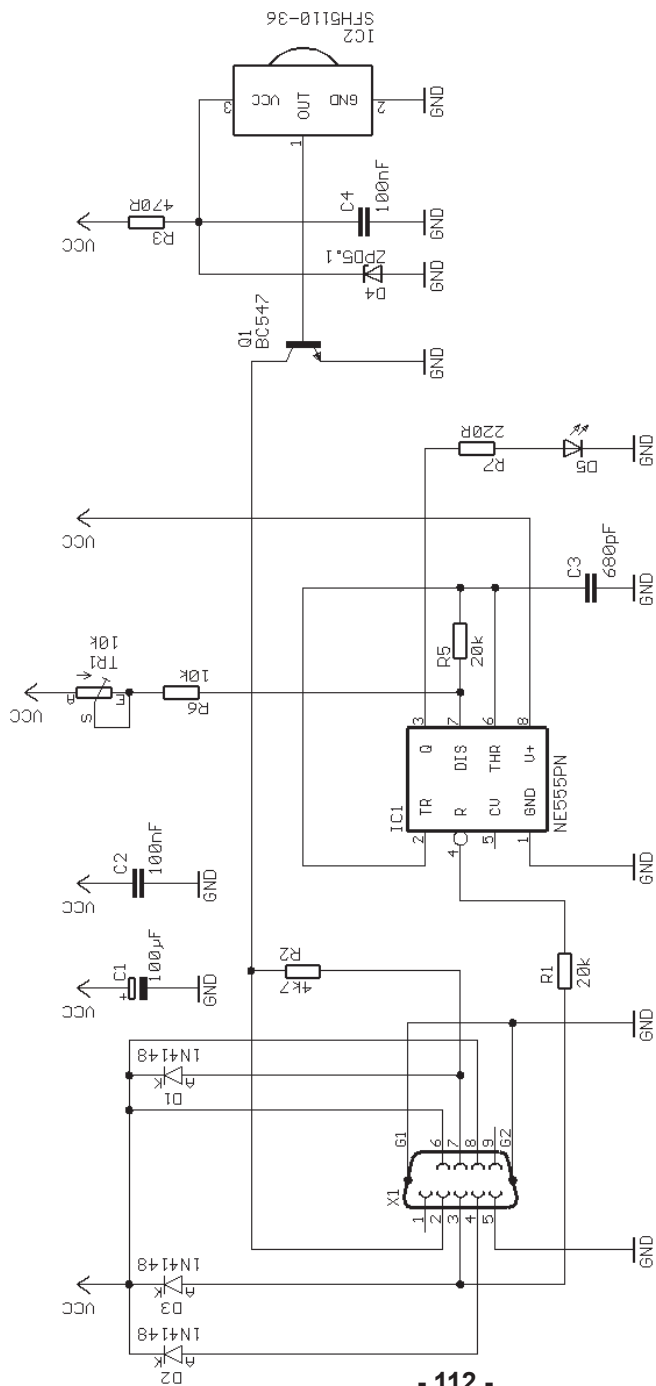
C. DIAGRAM DISPLAY MODULE



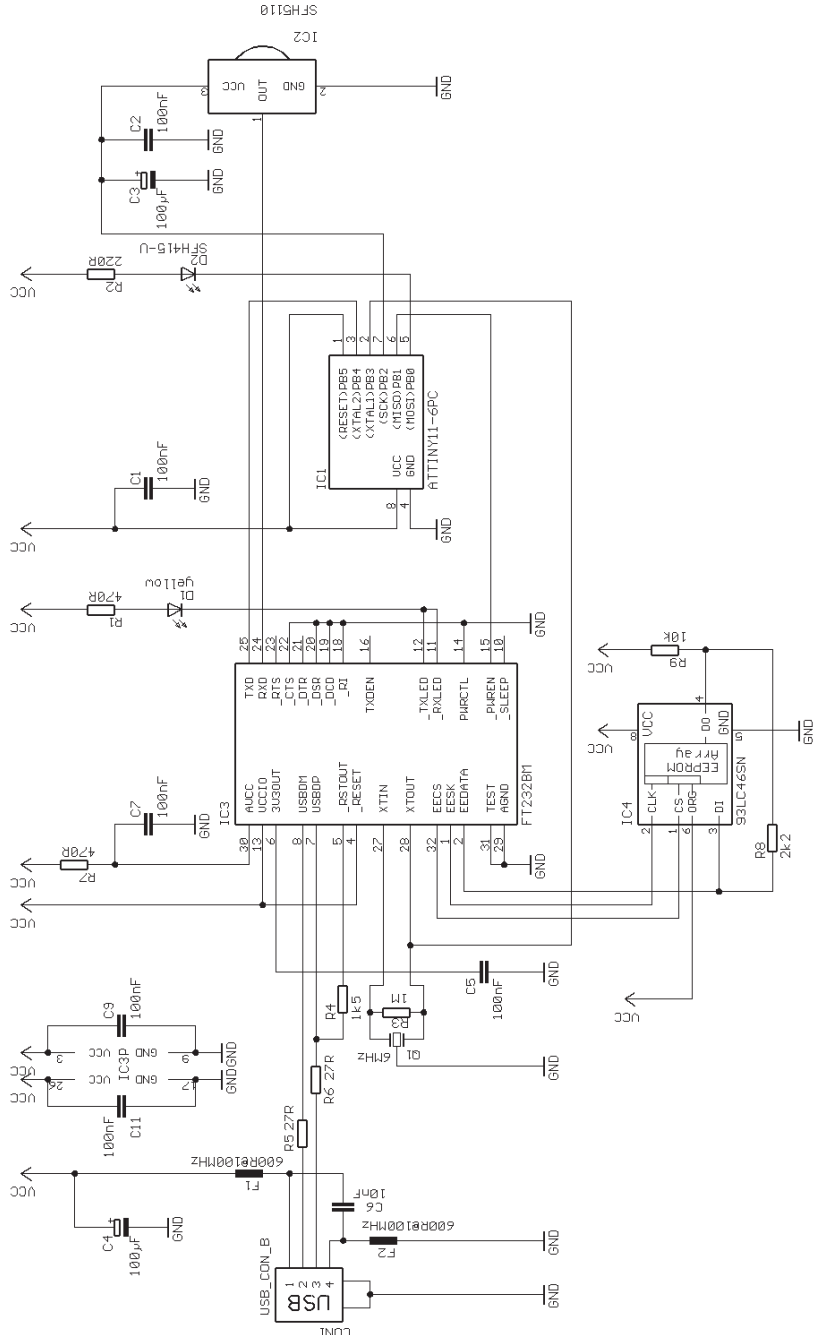
D. DIAGRAM ULTRASONIC MODULE



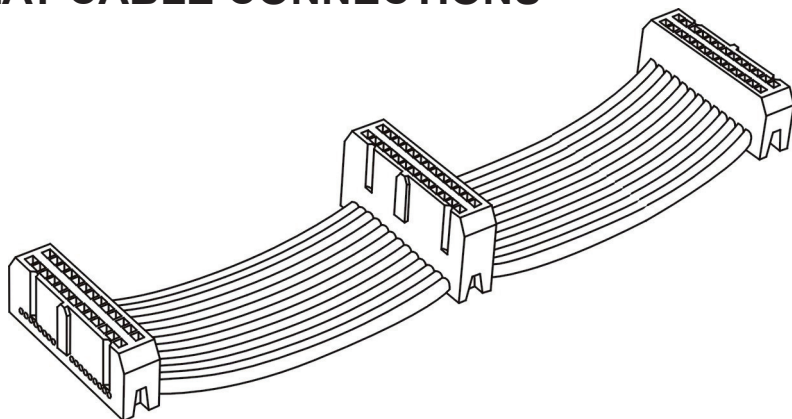
E. DIAGRAM RS-232 IR-TRANSCIEVER



F. DIAGRAM USB IR-TRANSCIVER



G. FLAT CABLE CONNECTIONS



Pin 1	SCL	Serial Clock (for I2C communication)
Pin 2	SDA	Serial Data (voor I2C communication)
Pin 3	PC3(ADC3)	Digital input/output or analog monitor input
Pin 4	PC2(ADC2)	Digital input/output or analog monitor input
Pin 5	PC1(ADC1)	Digital input/output or analog monitor input
Pin 6	PC0(ADC0)	Digital input/output or analog monitor input
Pin 7	GND	GND (several connectors to prevent signal noise)
Pin 8	GND	GND (several connectors to prevent signal noise)
Pin 9	AVCC	Analog reference-voltage for AD-converters
Pin 10	PC6(RESET)	Microcontroller reset pin
Pin 11	PB5(SCK)	Digital input/output
Pin 12	PB4(MISO)	Digital input/output or I2C function pin
Pin 13	PB3(MOSI/OC2)	Digital input/output or I2C function pin or Timer2 pin
Pin 14	PD3(INT1)	Digital input/output or external interrupt
Pin 15	PD6(AIN0)	Digital input/output or analog testinput
Pin 16	D7(AIN1)	Digital input/output or analog testinput
Pin 17	PD0(RXD)	Digital input/output or RS232 input
Pin 18	PD1(TXD)	Digital input/output or RS232 input
Pin 19	VCC	VCC
Pin 20	GND	GND (several connectors to prevent signal noise)

H. ERROR TRACING

H.1. GENERAL

Check all parts for correct polarisation and correct value. Check soldering connections for short circuits and bad soldering. Has a soldering pad been disrupted ?

If all checks have been made without results, the bad part has to be traced with the help of the schematic (see Appendix) and an adequate measurement device (multimeter or oscilloscope).

H.2. Failure of the RS232-IR-transceiver

- Activated key and displayed symbol do not match

Calibrate Trimmer TR1 until activated key and displayed symbol do match.

- The terminal program does not display any symbol

Has the timer-IC (IC1) been inserted and has it been inserted correctly polarized (the mark must point to the three diodes) ? Take any infrared remote control of a HIFI- or video-equipment (videorecorder, TV, etc) and point it to the IR-transceiver, pressing a few keys. If the terminal program displays irregular symbols, the receiver (IC2, R3, C4, D4, T1) is operating. All other parts have to be checked.

Still not working

(In most cases one of the components IC1, IC2, Q1, D4 may have been damaged).

H.3. USB-infrared-transceiver does not work

WINDOWS

Check if the drivers are installed correctly and if the correct com port is selected.

LINUX

Disconnect the USB transceiver... wait a minute and connect it again, this may solve the problem.

An other option is to install a new kernel

H.4. IR-interface

H.4.1. YETI does not send symbols

Check polarity of IR-Diode D10.

Check resistor R16 220Ω (red, red, brown, gold)

H.4.2. YETI does not receive symbols

You will need a line of sight between IR-Transceiver and ASURO (at a distance of max. 50 cm) and the IR-Transceiver must be checked and OK (see chapter [6.1](#)).

Check position and polarity of C2.

Check resistor R17 and C2.

470Ω (red,red, brown, gold)

100nF (imprint 104)

If you have not found the error yet, please remind the soldering of IC2. This component is sensitive to overheating and may have been damaged while soldering. In this case replace the component by a new IC (SFH 5110-36). When the transfer of data between PC and ASURO is malfunctioning again and again, re-adjust trimmer TR1 in the transceiver.

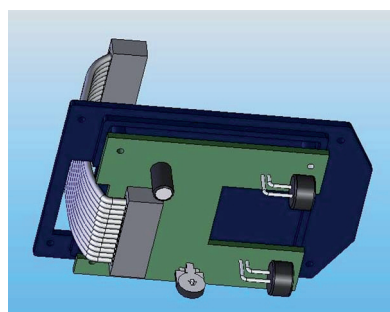
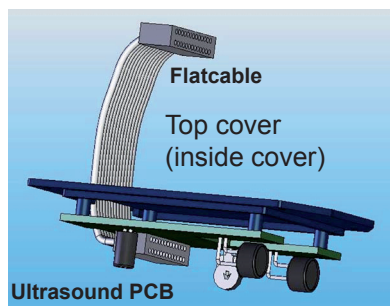
H.4.3. Things still do not work well

Check polarity of C8.

220μF/ at least 10V

If transfer of data between PC and YETI is malfunctioning again and again, re-adjust trimmer TR1 in the transceiver.

I. INSTALLING THE UPGRADE KITS



Installing Ultrasound PCB



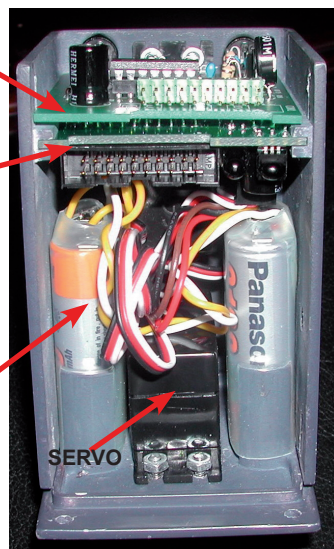
Installing experiment PCB

Ultrasound PCB

Main PCB

Accu Set

SERVO



Installing
Display PCB



J. CALIBRATION AND TEST SOFTWARE

We supply the YETI processor with a standard selftest and calibration program called 'test.hex'.

The Yeti 'test.hex' program contains:

1. a calibration mode
2. a walking mode

SELFTEST PROCEDURE

Switch Yeti on.

YETI BOOTLOADER STARTS

- both leds will immediately light up
- both servo's will turn clockwise a little bit
- after 3 seconds ...

#END OF BOOTLOADER PROGRAM

→ 'TEST.HEX' PROGRAM STARTS

'TEST.HEX' PROGRAM STARTS

- both leds go off
- both servos go to their (almost) centre position
- a series of beeps is heard
- the program now waits 3 seconds for any hyperterminal key to be pressed. If so, YETI goes into calibration mode. With the first hyperterminal key being pressed, YETI lets a short beep being heard once.

This beep indicates:

- that YETI now is in calibration mode
- that YETI has received an infrared signal, so it's infrared receiver is most probably okay.
- if any hyperterminal key was pressed, YETI is in calibration mode. Press the ENTER key in hyperterminal so YETI exits the calibration mode and starts with the walking mode.
- if no hyperterminal key is pressed, YETI will automatically exit the calibration mode after 3 seconds and continue with the walking mode

#END OF CALIBRATION MODE

➡ YETI'S WALKING MODE PROGRAM STARTS

YETI'S WALKING MODE PROGRAM STARTS

- a single beep is heard
- YETI's left LED turns on
- YETI leans to the LEFT and puts his RIGHT foot forward making 3 right/left steps
- YETI turns left, 4 steps
- YETI moves 3 steps backwards
- YETI turns right, 4 steps
- YETI shakes his body left/right 3 times
- YETI now keeps repeating above mentioned actions.
Between each action Yeti beeps and another LED combination is switched on or off

#YETI WILL STAY IN THE WALKING MODE FOREVER NOW

#REMARK ON ASSEMBLED YETI'S

It is important to check if YETI starts walking with his right foot first. Otherwise both servo connections may be connected to the wrong servo.

K. ACCU ADC VALUES

Calculation table ADC-Values for Accu voltage

Author: Uwiegw This EXCEL file can be downloaded at www.arexx.com

You can put the measured value at Pin21 (AREF) of the Mega8 internal reference voltage in the blue marked field.
This voltage should be approx. 2.56V, however can vary depending on chip tolerances, i.e. 2.7V.
From this reference voltage, the table calculates for every voltage value the ADC-value.

AREF: 2.71 <An Aref measured reference value.			
Accu voltage (in Volt)	ADC (V)	ADC-Value	Voltage range
Overload	6	2.73	1031 In this range the maximum operating voltage of some of the components is exceeded.
	5.9	2.68	1013 This can damage some of the components!
	5.8	2.64	996
	5.7	2.59	979
	5.6	2.55	962
Accu full	5.5	2.50	945
	5.4	2.45	927
	5.3	2.41	910
	5.2	2.36	893
	5.1	2.32	876
	5	2.27	859
	4.9	2.23	842
	4.8	2.18	824
	4.7	2.14	807
	4.6	2.09	790
Accu half full	4.5	2.05	773
	4.4	2.00	756
	4.3	1.95	739
	4.2	1.91	721
	4.1	1.86	704
Accu soon empty	4	1.82	687
	3.9	1.77	670
	3.8	1.73	653
	3.7	1.68	635
	3.6	1.64	618
Accu empty	3.5	1.59	601
	3.4	1.55	584
	3.3	1.50	567
	3.2	1.45	550
	3.1	1.41	532
	3	1.36	515
			Below this voltage the Mega8 will not function anymore.

This EXCEL file can be downloaded at www.arexx.com