AREXX Engineering

SCHOOL LAB

Walking robot
YETI ARDUINO

Manual: Model YT-5000

CONTENT

NOTICE!

Manufacturer:
AREXX Engineering
JAMA Oriental

European Importer:
AREXX Engineering
ZWOLLE, HOLLAND

Technical support:

WWW.AREXX.COM
WWW.ROBOTERNETZ.DE

AREXX Engineering

Impressum

©2015 AREXX Engineering
Nervistraat 16
8013 RS Zwolle
The Netherlands

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

E-Mail: Info@arexx.nl

Information about limited warranty and responsibility

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

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

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

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

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

Important safety recommendation

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

Symbols

This manual provides the following symbols:

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

Safety recommendations

- Prior to the assembly read the manual thoroughly.
- Be careful when using tools.
- Keep this kit away from young children during construction and operation. (They might get hurt by the tools or swallow small components).
- Check the polarity of the batteries or power supply.
- Keep all products dry, when the product gets wet remove the batteries or power directly.
- Remove the batteries or power when you are not using the product for a longer period.
- Children below 14 should only assemble this product with the help of adults
- Do not overload the motors.
- This module is equipped with highly sensitive components. Electronic components are very sensitive to static electricity discharge. Only touch the module by the edges and avoid direct contact with the components on the circuit board.

Normal use

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

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

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

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

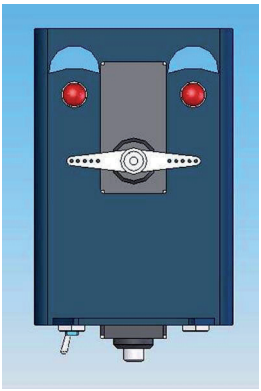
Please read all the safety instructions of this manual.
No return is possible after having opened the bags with components and pieces.

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:	ATmega328P
Programming language:	ARDUINO C++(Wiring)
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 tree extensions by a flatcable and stacking
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 abominable 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 the 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 Pulse 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 pulse 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.
- 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.
- Add self-made electronic extension units.
- Control YETI by wired or wireless signals from your computer or from your remote TV-control set.
- Let YETI play melodies or make sounds

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 ouverture.

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

With the USB IR-Transceiver

We will be using harmless invisible infrared light signals to load an example program into the YETI memory. In fact, the supplied USB-port adapter contains a USB-infrared transceiver, which must be connected to the USB-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.

With the USB cable

It is also possible to load programs into the Yeti with the extension kit and a USB cable, this will go much faster.

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 not a problem, as it may be reloaded from the PC into the memory at any time.

2.4. Loading an example program into the YETI memory

2.4.1. With the USB IR-Transceiver

We will be using harmless invisible infrared light signals to load an example program into the YETI memory. In fact, the supplied USB-port adapter contains a USB-infrared transceiver, which must be connected to the USB-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.

Place the USB-IR-Transceiver near the YETI IR-Transceiver, at a distance of max. 50 cm. The component sides of both PCBs must be facing and "seeing" each other. Now switch the YETI S1 to ON-position and start the UPLOAD in the ARDUINO software. When it does not work just pres UPLOAD again.

2.4.2. USB-Infraret-Tranceiver-Stick

First of all the IR-Transceiver must be installed first and tested. For this test connect the IR-Transceiver to a free USB port of your PC by a USB extension cable.Now a message will apear
"NEW HARWARE WAS FOUND": "*AREXX ASURO USB*"
Now you can install the FTDI USB driver from the YETI CD. When the driver is not detected automatically you can Download it from our website www.arexx.com or from the FTDI website directly. When the driver is installed you can approach the USB transceiver like a normal serial port.

2.4.3. Terminal test

After starting the terminal program (you first have to download an terminal program from Internet) f.e. "Hyperterminal", you will be asked to define a name for the connection. You may choose YETI USB 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

This USB-IR-transceiver-stick suppresses its own echo. This is a clear advantage in data transmission but the self test via reflection in the HyperTerminal doesn’t work anymore. ***The proper installation of the driver can be checked by the fact that the green transmission diode flashes during data transmission or typing on the HyperTerminal.*** The functions of the USB IR transceiver have been checked in the factory and a default is therefore hardly probable.

When it transmits, the yellow LED flashes and when it receives, the green LED is on. The transceiver won’t read its own echo and therefore you need YETI during its self test to test it.

If you do not have any success in this procedure, we do have a problem with the circuit, which should be solved.

2.4.4. USB-infrared-tranceiver does not work

Check if the drivers are installed correctly and if the correct com port is selected. 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. Check with any digitale camera (phone camera) if the IR LED lights up.

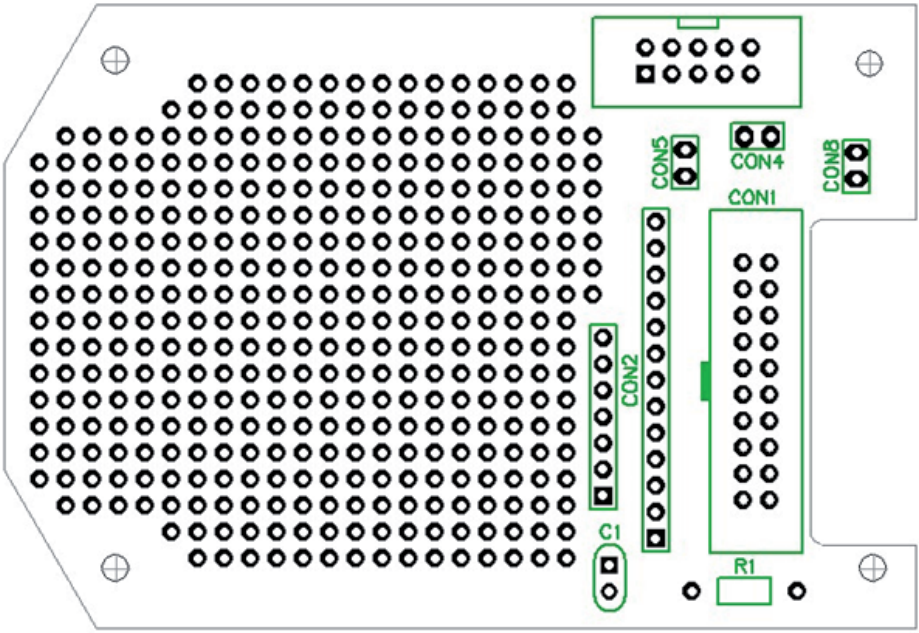
2.4.5. With the USB cable

It is also possible to load programs into the Yeti with the extension kit and a USB cable, this will go much faster.

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 not a problem, as it may be reloaded from the PC into the memory at any time.

YETI Programming extension kit YT-PRG

The YT-PRG is an optional extension for the YETI ARDUINO walking robot which is available separately see chapter 14.11



2.5. Loading a YETI program into the robot

2.5.1. Wired uploading

- Connect the USB adapter to your PC.
- Start the Arduino IDE
- Select the COM-port for the USB-port adapter in the Arduino program on the computer.
- Select a Yeti program in the Arduino IDE, can be done from File -> sketchbook
- Connect the wired expansion board to your Yeti, either plug it onto the display or into the flatcable (if it already was connected, you do not have to take it off before this step).
- Connect the USB adapter to Yeti through the smaller, but longer
f latcable with 10 pins.
- Switch Yeti on
- Press the “Upload” button in the Arduino IDE
- The program is now being compiled and transferred to Yeti’s memory.
- Wait until the Arduino IDE is done uploading.
- 5 Seconds after completion Yeti will start to execute the program.

2.5.2. Wireless uploading

- Connect the USB adapter to your PC.
- Start the Arduino IDE
- Select the COM-port for the USB-port adapter in the Arduino program on the computer.
- Select a Yeti program in the Arduino IDE, can be done from File -> sketchbook
- 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 “Upload” button in the Arduino IDE.
- Switch on YETI’s main switch (if compiling takes very long, retry uploading but turn Yeti on when the status bar reads: “uploading”).
- Now the YETI program will be transferred into the YETI processor memory.
- Wait until the YETI program has been loaded.
- 5 Seconds after completion Yeti will start to execute the program.

- If something timed out or gave an error, see the next page.
- Connect the USB adapter to your PC.
- Start the Arduino IDE
- Select the COM-port for the USB-port adapter in the Arduino program at the computer.
- Select a Yeti program in the Arduino IDE, can be done from File -> sketchbook
- 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 "Upload" in the Arduino IDE.
- Switch on YETI's main switch (if compiling takes very long, retry uploading but turn Yeti on when the status bar reads: "uploading").
- Now the YETI program will be transferred into the YETI processor memory.
- Wait until the YETI program has been loaded.
- Wait for roughly 5 seconds.
- If you upload using the wired connection, you do not need to turn of Yeti before pressing the upload button, Yeti will reset automatically.

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

2.7. The communication between YETI and PC

Pressing the "Upload"-Button in the Arduino IDE will activate the Arduino program to contact the YETI robot. If YETI responds, the YETI program will be transferred. YETI however will only respond to the PC if the contact is tried within a 5 seconds period following YETI's switch on. If YETI is not being contacted within the 5 seconds period, the robot will proceed by starting the program in its memory. In case the YETI does not respond, the Arduino program will output an error message.

Reports from ASURO-users describe potential problems in data transfers.

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 and sun light (especially fluorescent lamps).

2.7. 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. ARDUINO General Description

2.1. Who or what is ARDUINO?

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

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

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

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

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

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

The English version of the name is "Hardwin".

2.2 Microcontrollers!

2.2.1. Applications

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

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

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

2.3. Power Consumption and Speed

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

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

2.4 Microcontroller Programs

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

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

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

2.5. Interface Architecture

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

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

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

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

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

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.

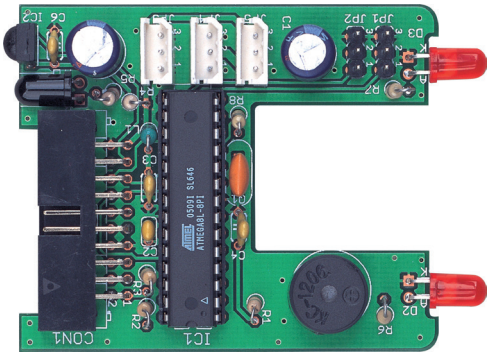
At this point the legs servo is activated. The legs servo moves YETI's left leg forward as far as possible. While moving the left foot forward from behind, the body will also be bended up to a position halfway the left foot, but still following the line of the right foot. YETI is still balancing on his 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.

5. HARDWARE

5.1. YETI main PCB

YETI's main printed circuit board (PCB) is equipped with an Atmega328P 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.



5.2. 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.

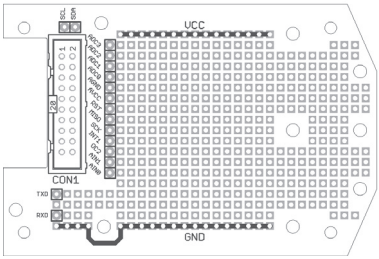
All remaining flat cable connections are connected to a dedicated microcontroller pin. The pin numbers are listed with the assigned microcontroller pin functions.

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

5.3. 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



5.4. YETI experimental extension set

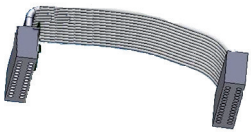
Summary
The experimental extension set has been designed to set up your own electronic designs and connect the experimental modules to the micro-controller.

5.5. 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

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

PCB

YETI Pins Component side

PCB

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

Cable

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/SCK	PB4/MISO	RESET	AVCC	AGND	PC0/ADC0	PC1/ADC1	PC2/ADC2	PC3/ADC3
Upgrade kit	US		US							US			

Con3 I2C

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

Con4 UART

	1	2
	17	*18*
	PD1/TXD	PD0/RXD
	IR	IR

Con5 VCC

	1	2
	20	*19*
	GND	VCC

6. ASSEMBLY INSTRUCTIONS ELECTRONICS

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

6.1. 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 dificult 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

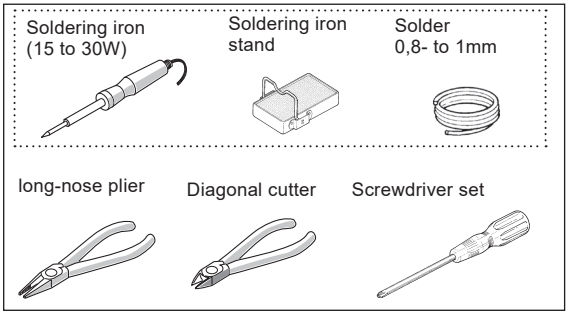
CAUTION

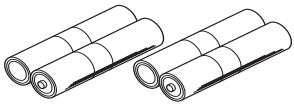
- Read this manual carefully in advance to fully understand how to assemble this product.
- Children below 14 should can only assemble this product with the help of adults.
- Be careful about tools. Especially be careful about sharp tools such as nippers or cutter knife to prevent any injuries or accidents.
- 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 recharchable and non rechargeble 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.

6.2.Necessary tools

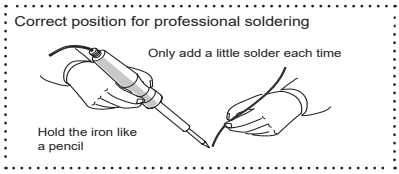


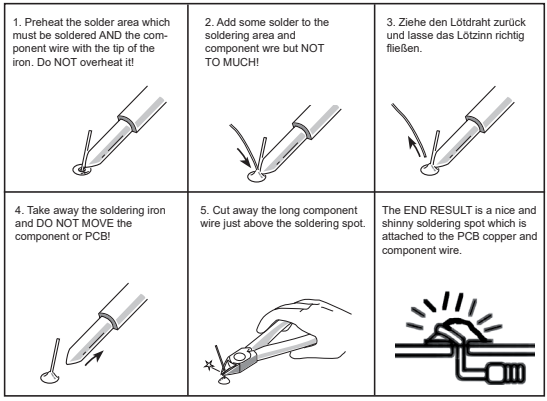


Necessary Batteries:
AA Batteries, 3 Pieces
(not included)

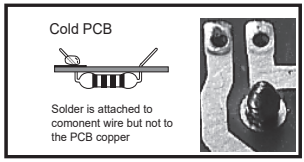
6.3. Soldering techniques:

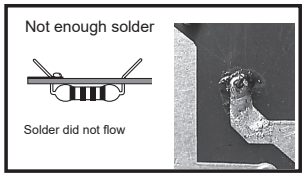
Only use lead free ROSIN CORE solder!
Never use any liquid- or paste flux!

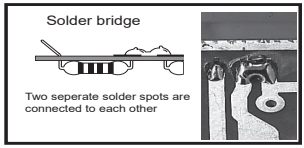


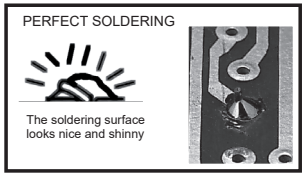


5.4. Troubleshoot soldering mistakes:









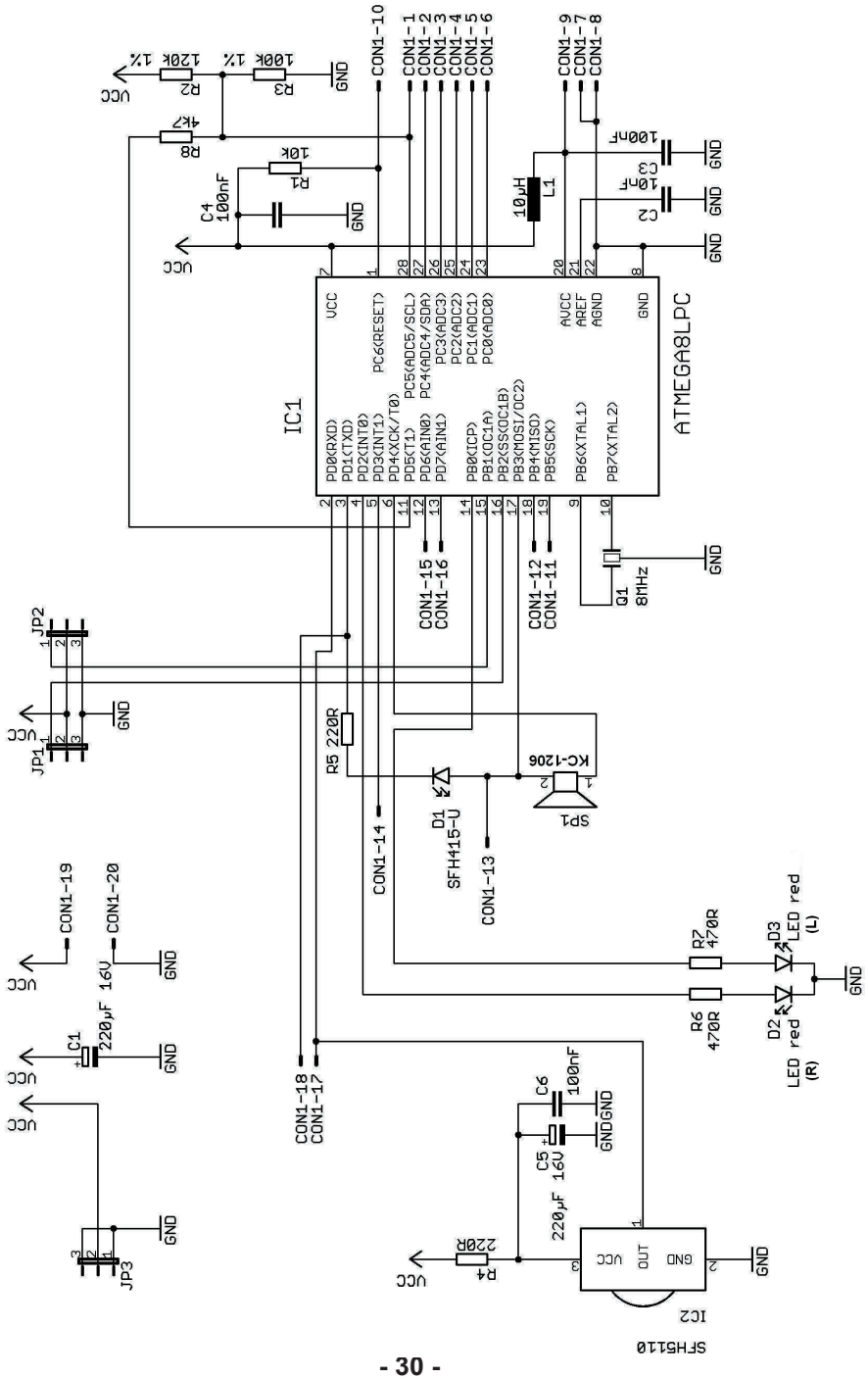
6.4. Assembly main PCB

IMPORTANT see 5.5 (Diagram) and 5.6 (Pictures)

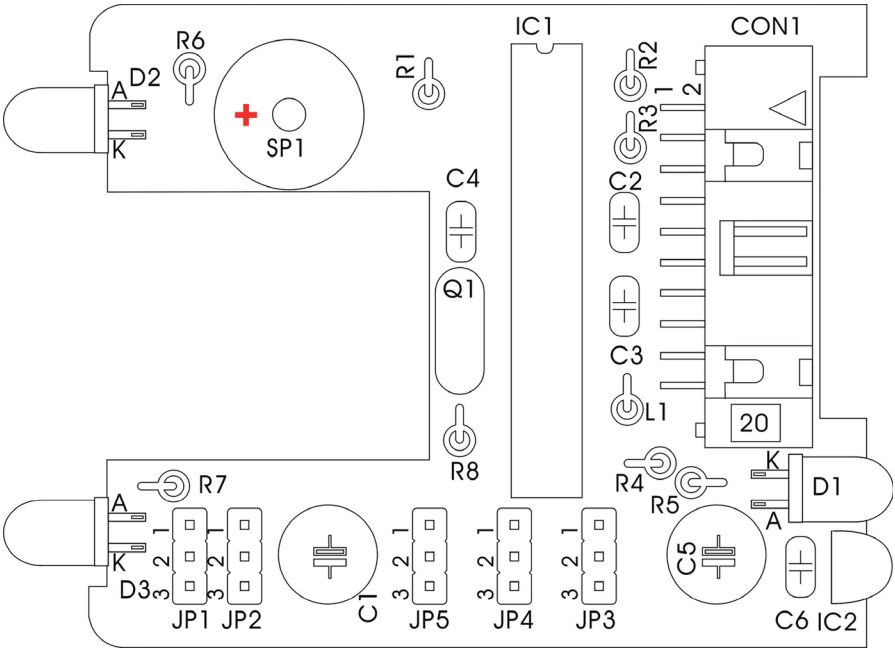
YETI parts list

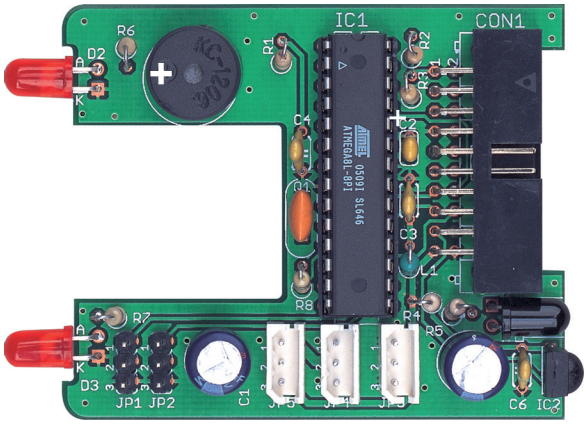
Nr.	Name	Pcs.
PCB1	Main PCB	1
IC1	ATmega328-P-PU *(polarity!)*	1
IC2	SFH5110 IR-receiver-IC *(polarity!)*	1
R1	10K / 0.25W / 5% *(Brn, blk, or, gld)*	1
R2	120K / 0.25W / 1% *(Brn, red, blk, or, brn)*	1
R3	100K / 0.25W / 1% *(Brn, blk, blk, or, brn)*	1
R4	220R / 0.25W / 5% *(Red, red, brn, gld)*	1
R5	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	Resonator, 16Mhz / 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

6.5. Diagram YETI



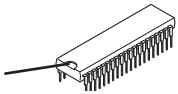
6.6. Hauptplatine





IC

IC Marking

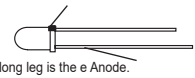


Resistor & Coil

Marking with color stripes

LED & IR-LED

The side with the flat marking is the Cathode.



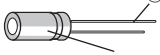
The long leg is the e Anode.

Condensator

No polarity

ELCO (Elektrolyt Condensator)

The long leg is the ⊕



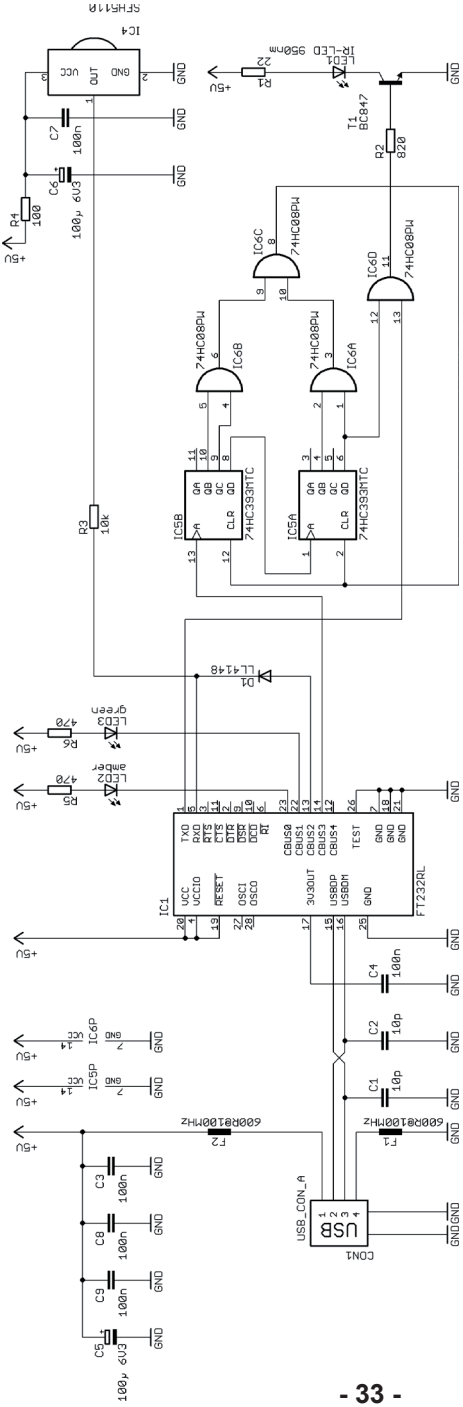
The white line marking on the housing is the ⊖

6.7. 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: 100µF 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: 20kΩ_ (red, black, orange, gold)
- R2: 4.7kΩ (yellow, violet, red, gold)
- R3: 470Ω (yellow, violet, black, gold)
- R4: Does not exist
- R6: 10kΩ (brown, black, orange, gold)
- R7: 220Ω (red, red, brown, gold)
- TR1: 10KΩ 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!

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

6.8. Diagram USB IR-Transceiver



6.9. Layout USB-Infraread-Transceiver PCB

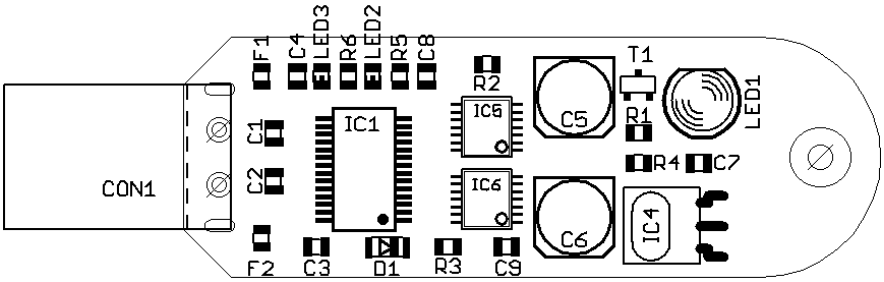


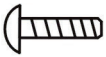
Fig. 5.1.: Layout USB- Infrared-Transceiver

7. PARTS LIST MECHANICS

Nut M3

O 8 pcs.

Flathead screw M3x8mm



O 8 pcs.

Collar big

O 4 pcs.

Colar small

O 4 pcs.

Collar screw

O 10 pcs.

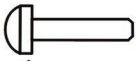
Nut M2

O 10 pcs.

Spacer

O 4 pcs.

Rivet



O 4 pcs.

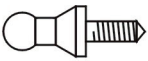
Screw-rod connector

O 2 pcs.

Servo Screw

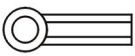
O 2 pcs.

Ball-end screw



O 4 pcs.

Ball adjuster



O 4 pcs.

Ballhead screw

O 4 pcs.

HEX-tool

O 1 pc.

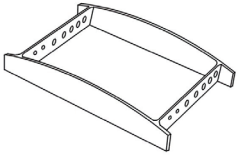
Thread end rod long

O 2 pcs.

Thread end rod short

O 4 pcs.

Foot

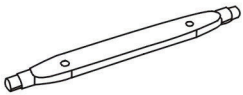


O 2 pcs.

Rod
5 x 80mm

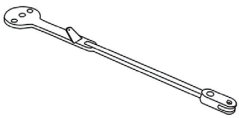
O 2 pcs.

Feet joint panel



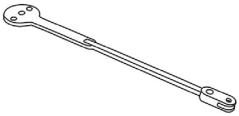
O 2 pcs.

Front leg



O 2 pcs.

Rear leg

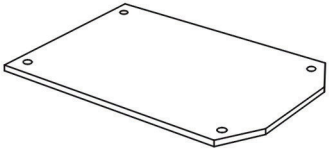


O 2 pcs.

Servo arm pushrod

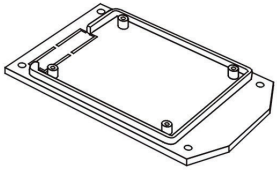
O 2 pcs.

Top cover panel



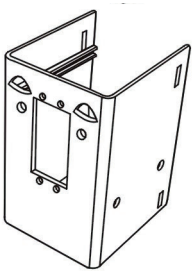
O 1 pc.

Top panel



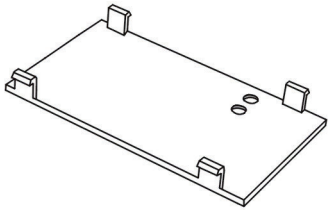
O 1 pc.

Head panel



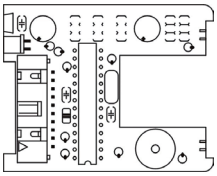
O 1 pc.

Back panel



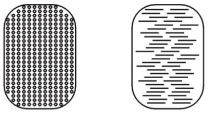
O 1 pc.

YETI main PCB (assembled)



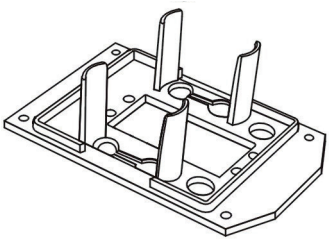
O 1 pc.

Velgro-tape



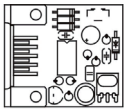
O 2 pcs. Male
O 2 pcs. Female
Pre-assembled

Bottom panel



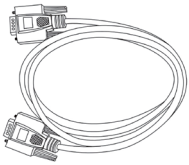
O 1 pc.

IR/Transceiver PCB (assembled)



O 1 pc.

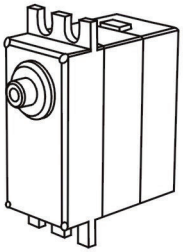
RS-232 Cable



O 1 pc.

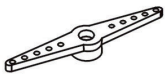
7.1. Important electronic parts

Servo motor



O 2 pcs.

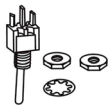
Servo arm



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

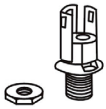
Pre assembled with cableset

Switch



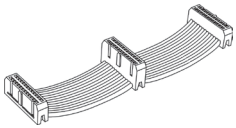
O 1 pc.

DC-connector



O 1 pc.

Flat cable

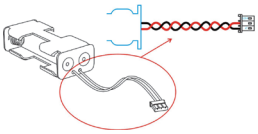


O 1 Pc.

Cable set, Pre-assembled

O 1 pc.

Battery holder



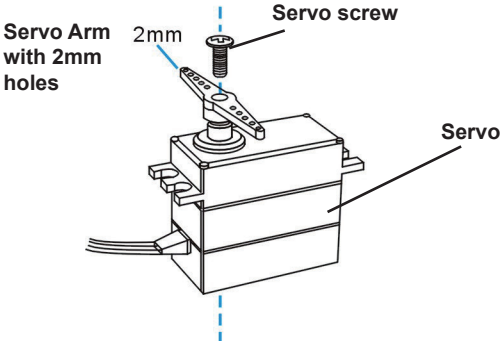
O 2 pcs.

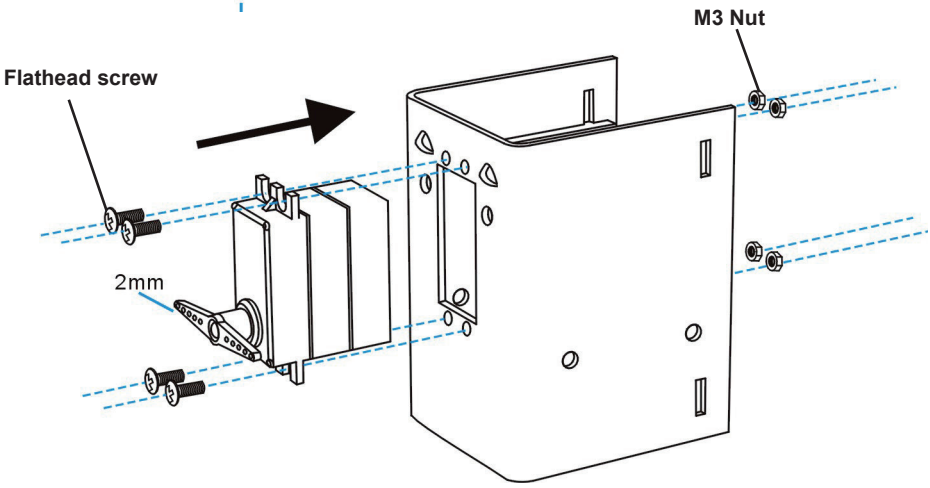
8. INSTRUCTIONS MECHANICAL ASSEMBLY

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





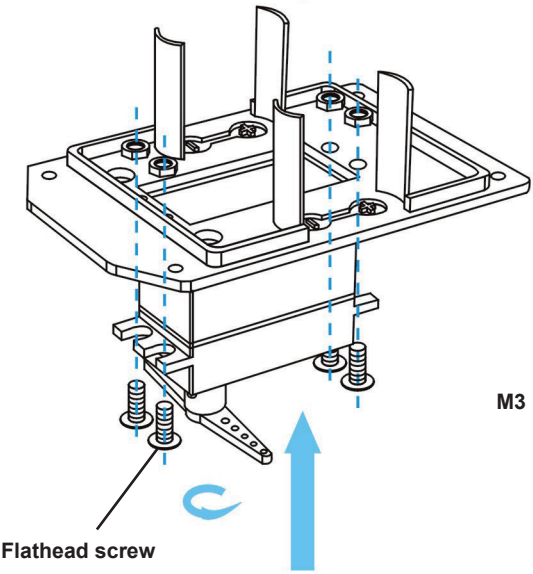
Assemble the servo as shown in the drawings.

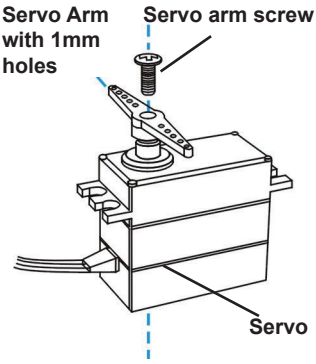
Assemble the servo arm to the servo, see the detailed drawing!

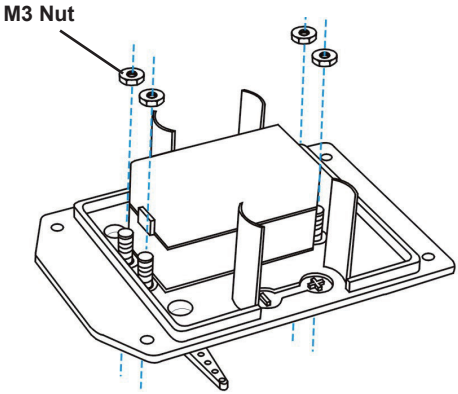
Installing the bottom servo:

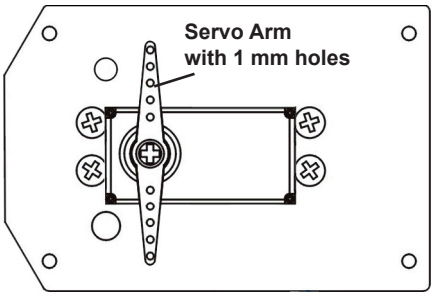
For this assembly you need;

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









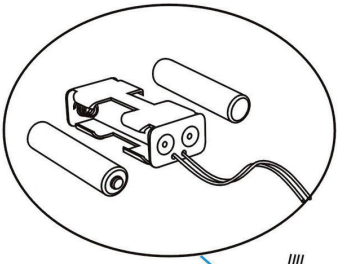
Assemble the servo as shown in the drawings.

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

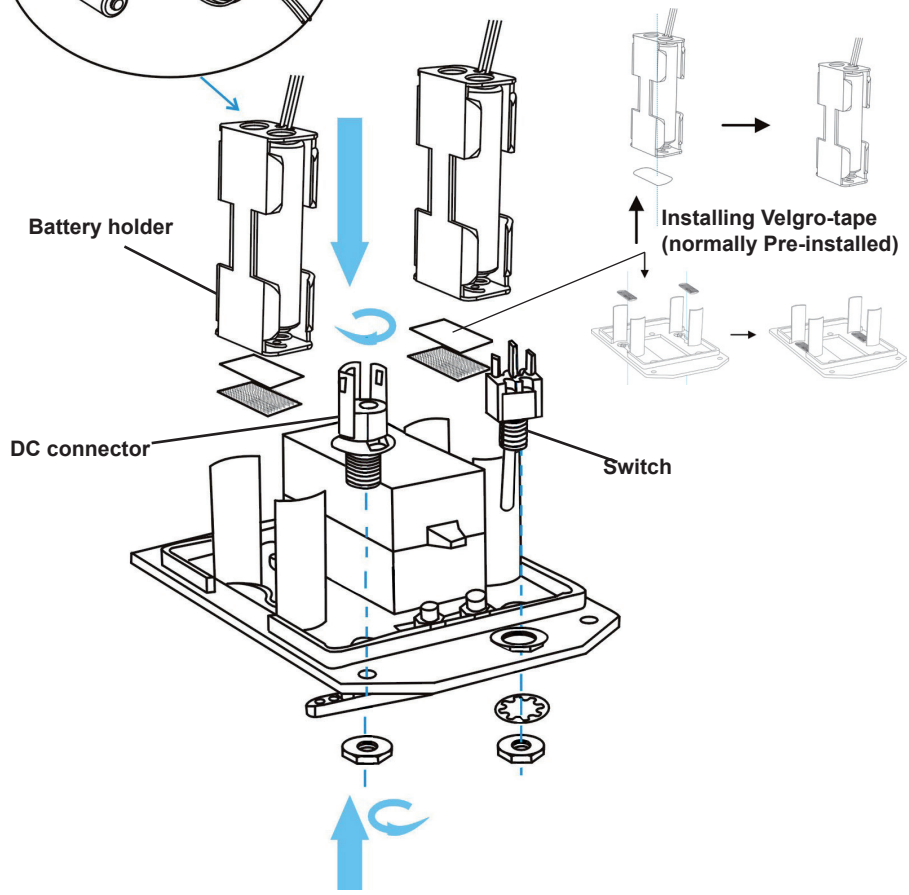
End assembly bottom panel:

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



IMPORTANT!
Do not forget to insert the batteries before you close the YETI back panel!

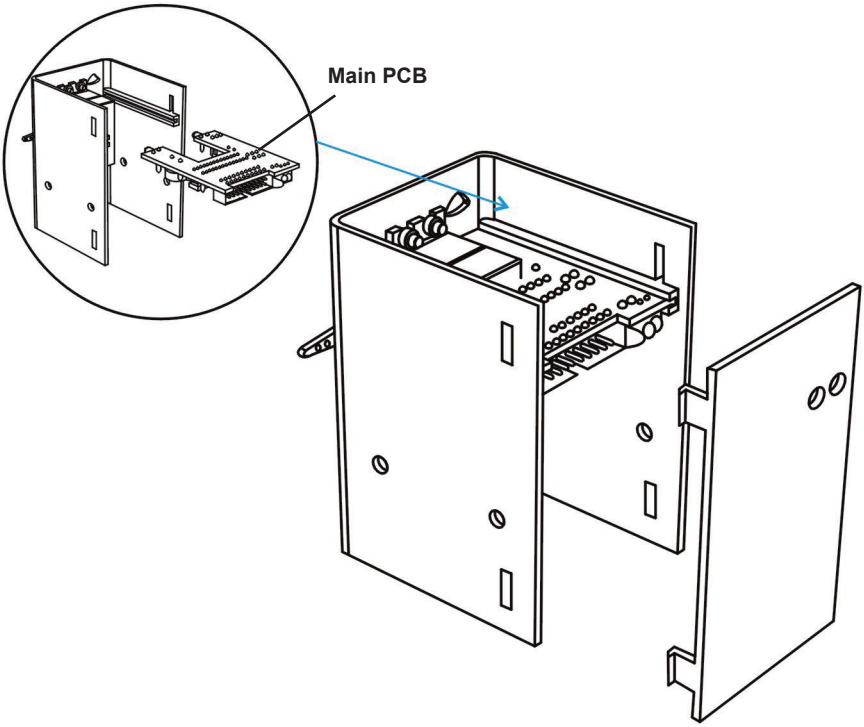


Assemble the bottom plate as descriped in the drawings.

Assembly YETI head:

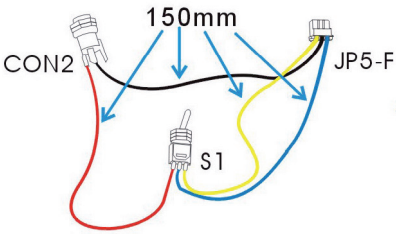
For this final assembly you need;

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



IMPORTANT!

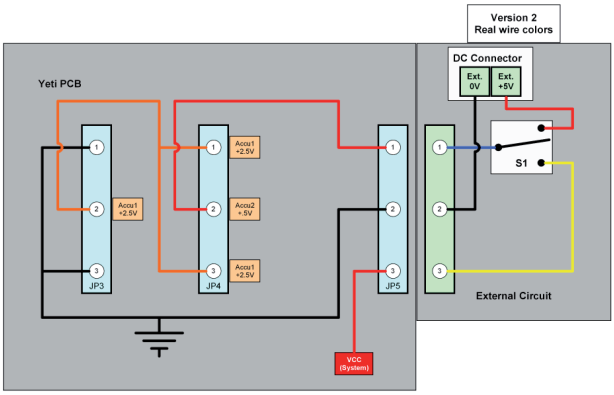
FIRST, before closing the head, you have to install all the wiring and cable sets! See page 33 and 34



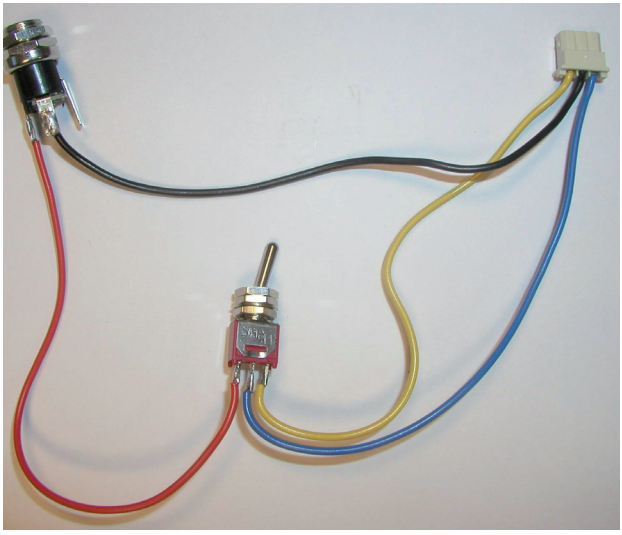
Installing the wires:

For the wire assembly you need;

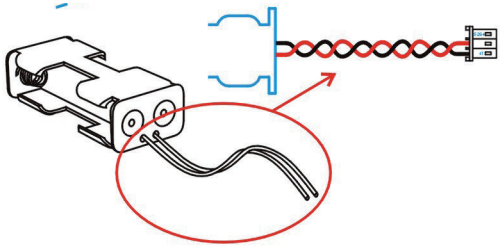
1 pc. Assembled bottom panel
1 pc. Assembled head panel
1 pc. Assembled main PCB
Assembled cable set with:
1 pc. Wire back
1 pc. Wire blue
1 pc. Wire yellow
1 pc. Wire red



Install all wiring as shown in the drawings.



Assembled wire set
Normally pre-assembled

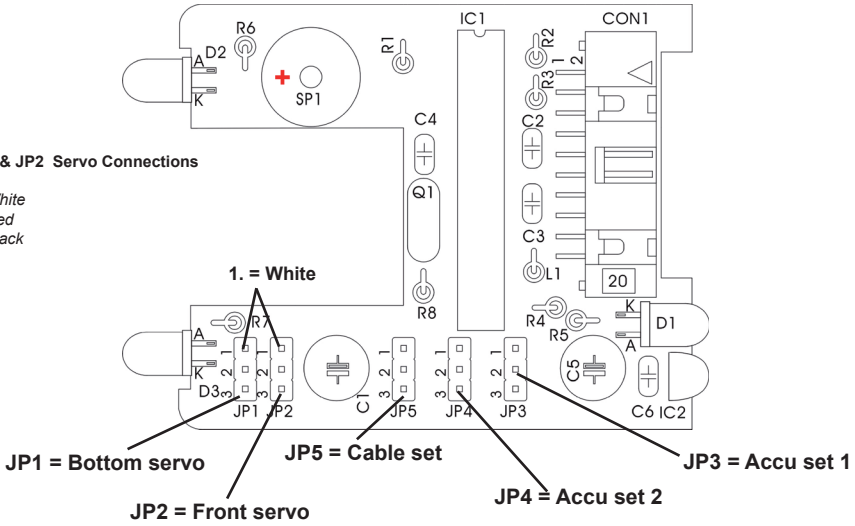


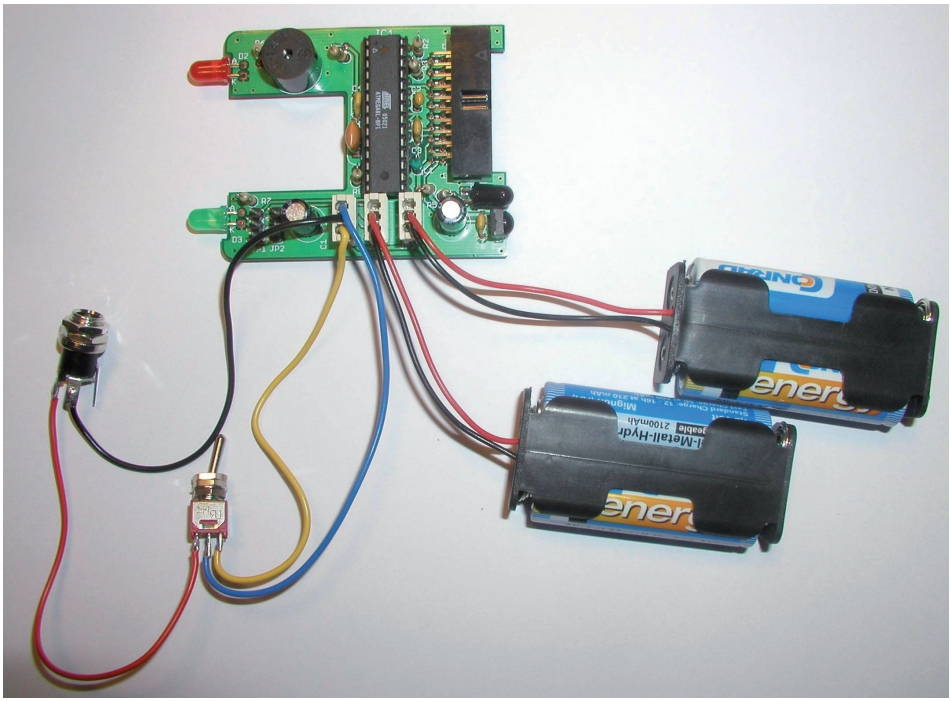
Assembled
battery holder

Cable connections main PCB:

JP1 & JP2 Servo Connections

1. White
2. Red
3. Black

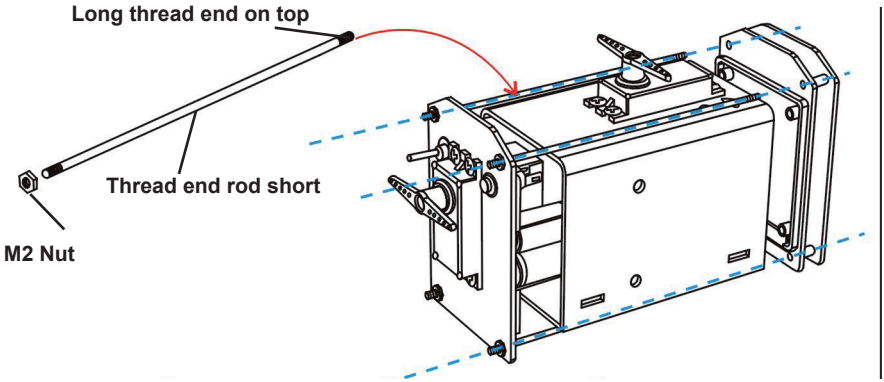


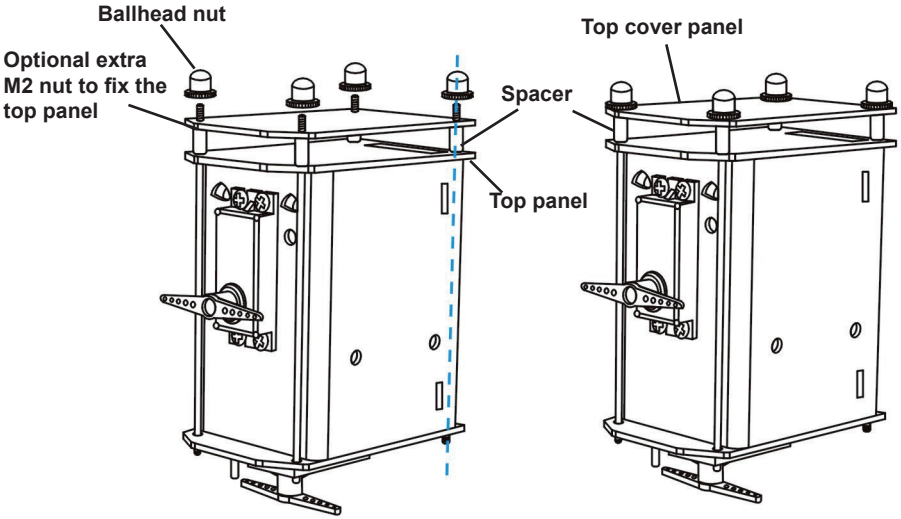


Mechanical end assembly YETI head:

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



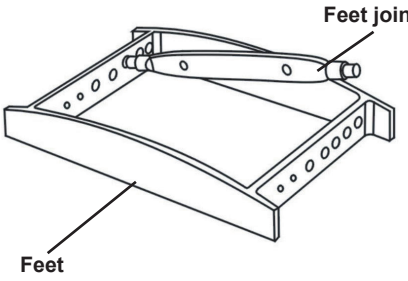


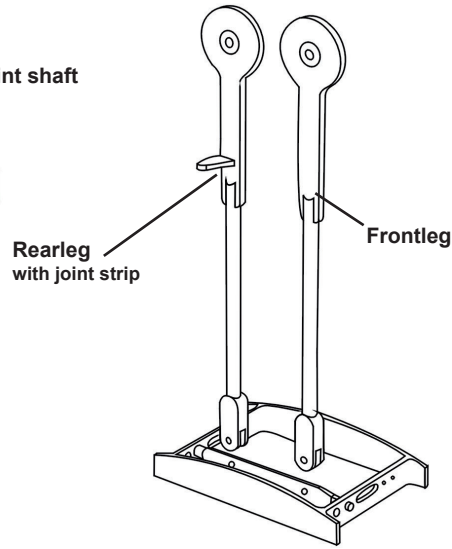
Legs and Feet assembly:

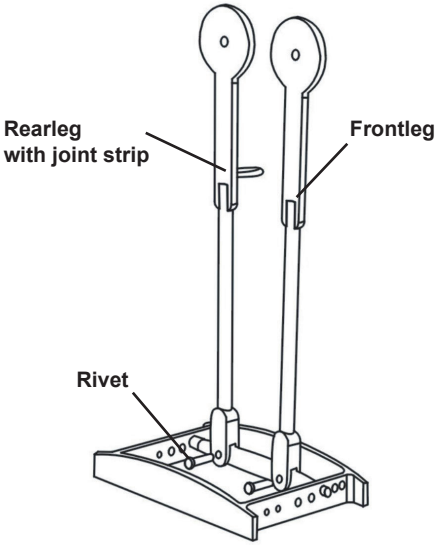
For the legs and feet assembly you need;

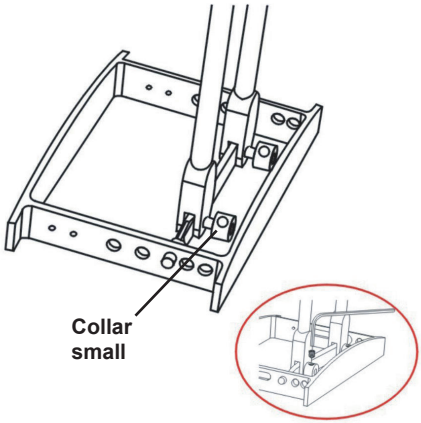
2 pcs. Foot
2 pcs. Front leg
2 pcs. Back leg
2 pcs. Feet joint shaft
2 pcs. Rivet
2 pcs. Collar, small

Assemble the feet and legs exacty as shown in the drawings







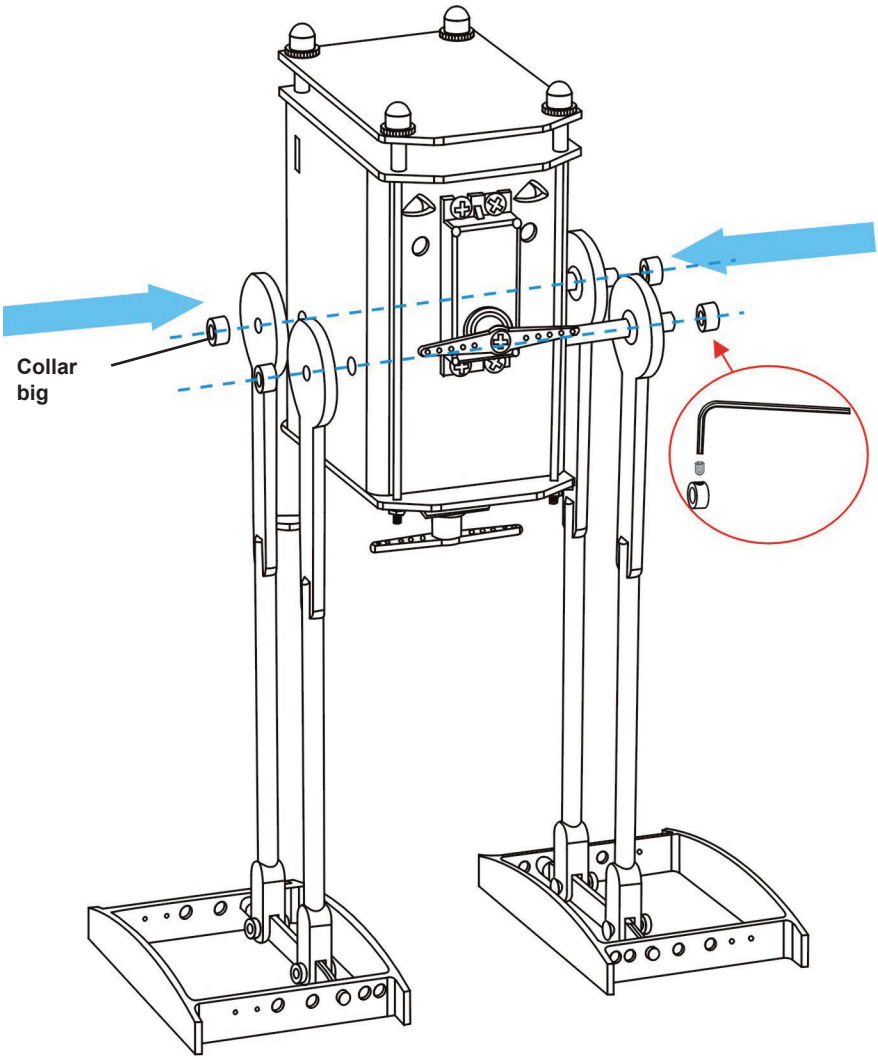


YETI Leg Assembly part I:

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

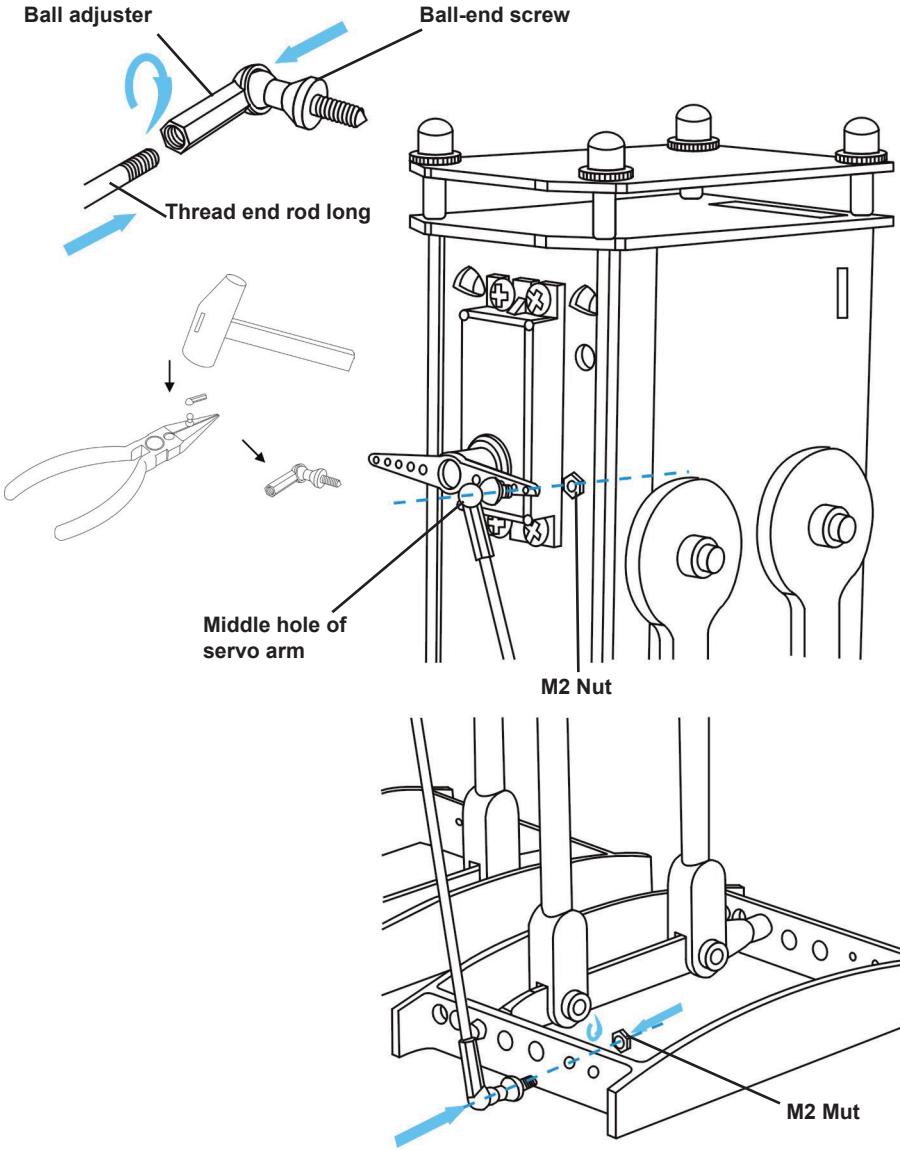


YETI Leg Assembly part II:

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

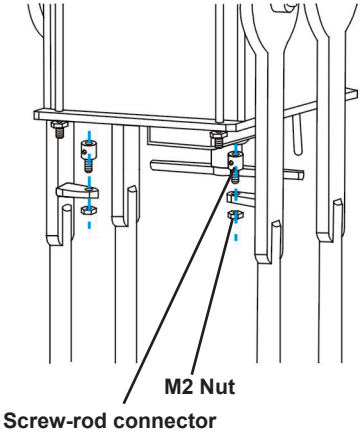
Assemble the servo rods exactly as shown in the drawings



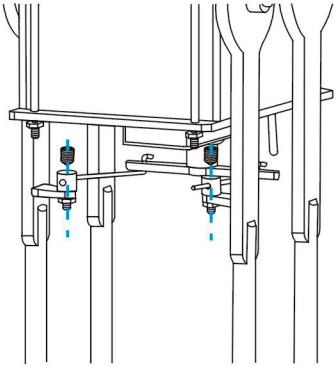
Final assembly YETI legs:

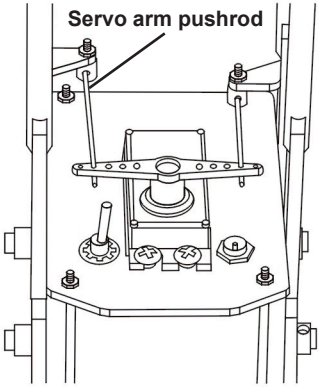
For the final leg assembly you need;

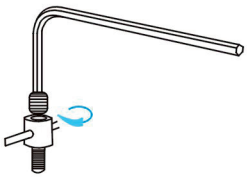
1 pc. Assembled Chassis
2 pcs. Nut M2
2 pcs. Servo arm pushrod
2 pcs. Screw-rod connector



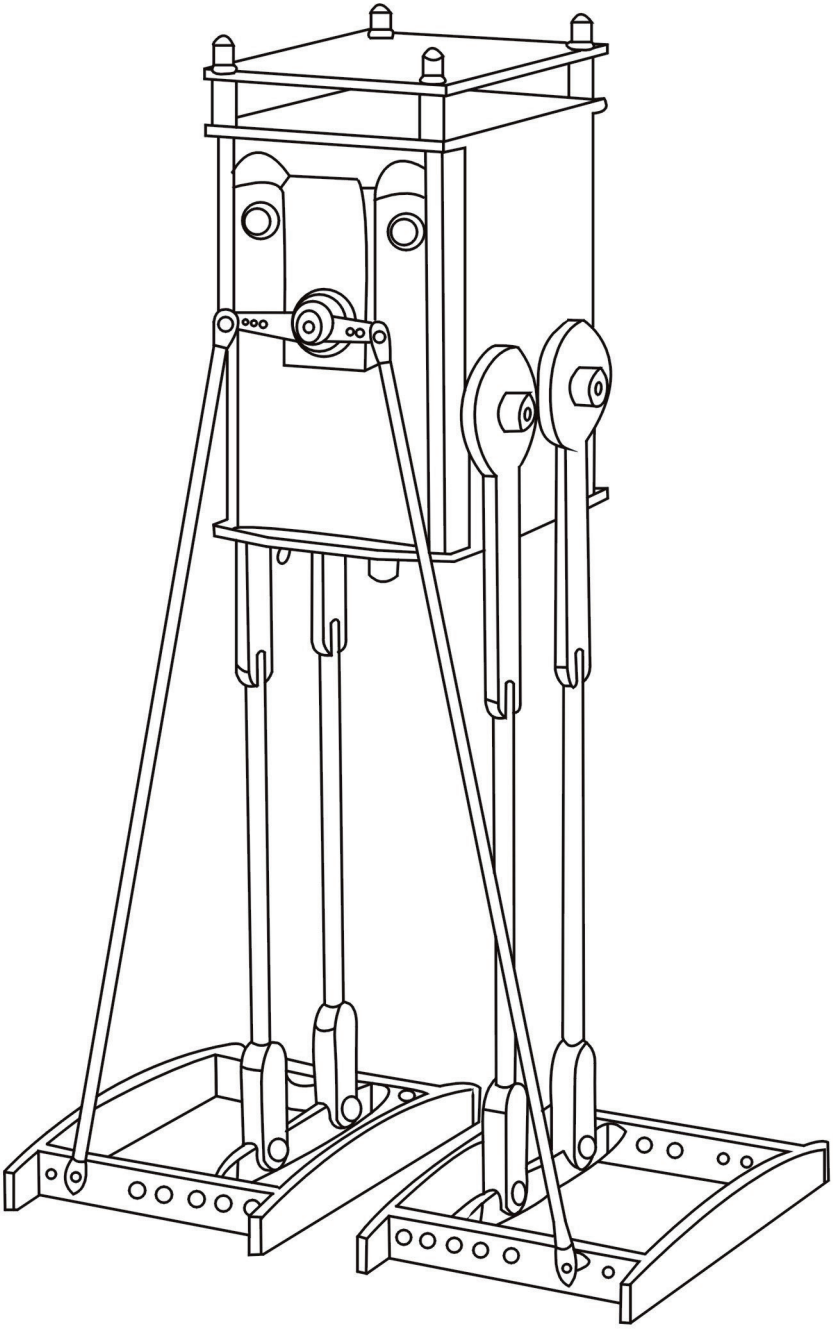
Assemble the servo to the rod as shown on the drawings







YOUR YETI IS READY !



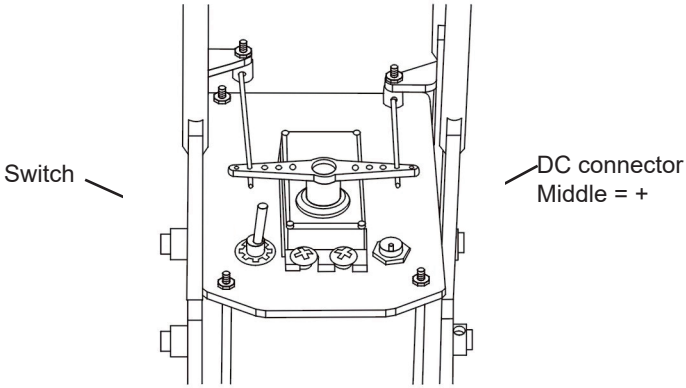
9. CHARGING THE YETI BATTERIES

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.



Charching YETI batteries:

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

See also APPENDIX K

10. SOFTWARE INFO

10.1. 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: “demonstration.ino, see example folder in the Yeti library”)
- Transfer the program to the YETI
  You will need just two 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.

10.2. 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 popular programming language called “C++”. You will need a special word processor (an editor) to write the YETI program. The editor we are going to use is built into the Arduino program

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 Arduino program is providing some features to help you in editing and trouble shooting in various programming languages. The Arduino IDE will help you by highlighting all
special commands, comments, functions and variables in various colors. These features will be a great help in programming.

10.3. Step 2 Compiling a “YETI”-program

Now comes the easy part, uploading the program to the Yeti. In the previous version of Yeti we actually had a step before we could upload the program, but for this version we have greatly simplified the process.

The Yeti has 32kiloBytes of flash memory on board, this is the memory where the program is stored. For us to use there is around 30kiloBytes of room left, this is because Yeti contains a hidden piece of code called the “bootloader”. This part of code allows us to make the uploading part so much more simple. The bootloader cannot be edited, read or deleted by user actions.

Switching Yeti on will always first start the bootloader, before it starts the user made program it stays in the bootloader “mode” for around 5 seconds. In these 5 seconds it listens to the Serial bus to see if a user is attempting to upload a program, either wired or wireless. Based on whether the user has installed the programming expansion shield and if a connector on that board is bridged, it selects between high and low speed. It has to do this because wireless does not support high speed serial communication.

Once Yeti has detected a user trying to upload a program it reads data from the Serial bus and writes it to its internal flash memory.

11. 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 copy the program on your hard drive first. To copy 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 copying the following will need to be done:

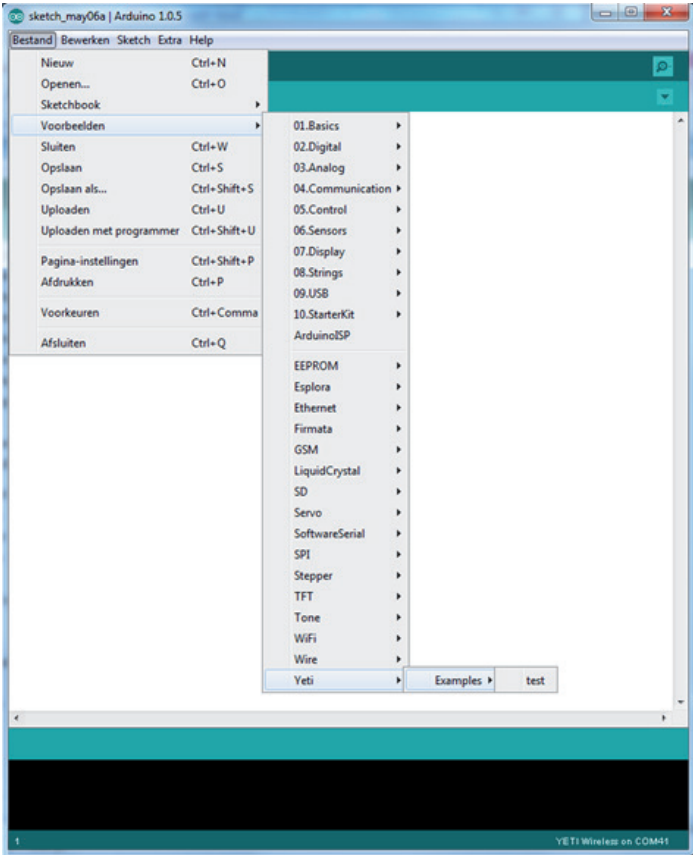
1. Copy the folder named Arduino to C:\Program Files\

11.1. ARDUINO IDE

In the Arduino installation folder (C:\Program Files\Arduino\) right click on the Arduino executable and make a shortcut to your desktop. Click on the shortcut to start Arduino. As soon as the Arduino IDE has started, which might take some time if it's the first time, click on the tab named "Extra". Navigate to the "boards" tab, if you are using the wireless adaptor select YETI Wireless, otherwise select YETI Wired High-Speed.

For more information about the Arduino IDE, please visit this page:
http://www.arduino.cc/en/Guide/HomePage

To try one of the example programs, navigate to the “File” tab and select “Examples”, down in this menu you can select “Yeti” and click “Examples”, see the picture below.



In the Extra tab also select the correct COM port, if there is just one, pick that one (Yeti programmer (wired/wireless) has to be plugged in). If there are multiple options, navigate to the Windows start menu, right click on “Computer” and select “manage”. Navigate to device manager and select “Ports (COM & LPT) and look for a name like “USB Serial Port(COM xx)” or “YETI Programmer”.

The buttons on the menu in the top of the window correspond to the following actions:

 Verify

Checks your code for errors.

 Upload

Compiles your code and uploads it to the Arduino I/O board. See uploading below for details.

 New

Creates a new sketch.

 Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File --> Sketchbook menu instead.

 Save

Saves your sketch.

 Serial Monitor

Opens the serial monitor, this can be used by your program to send data back to the computer

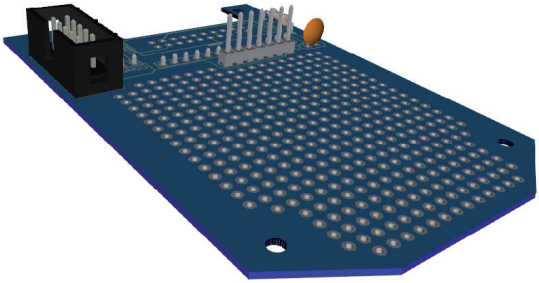
If you have not yet selected an example sketch (Hello World is recommended as a first program), select one now. Connect either the Yeti Wireless adaptor to your pc or the wired programmer to both your pc and to Yeti. To program Yeti, press the Upload button, after you have pressed that, switch on Yeti. Don't wait too long with switching on Yeti. If you have a wired connection Yeti may be powered up before pressing the button, it resets automatically. If you get an error, check: proper stable USB connection, if you have the correct COM port, if Yeti's batteries are full, if cables are in good condition and if the connection is wireless the distance between Yeti and the programmer may be too large or there might be too much ambient light. The wireless programmer is disrupted by the sun so please shield it. As for the wired programmer, make sure the boot loader is set to the wired version and if CON8 on the programmer board on top of YETI is bridged. CON8 sets the upload speed, bridged sets the speed to 57600 Baud, not bridged or no programmer extension board sets the speed to 2400 Baud.

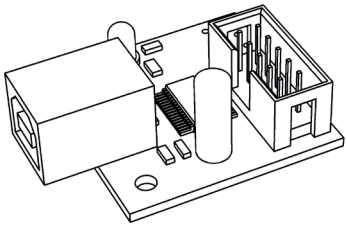
12. TEST AND OPERATION

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.

12.1. Wired USB to serial converter

Using the wired USB to serial converter is the most stable and fastest way to upload programs to Yeti, it also allows for feedback by Yeti by using the built in serial port viewer in the Arduino IDE. The button for the serial port viewer is in the upper right corner, see symbol explanation on the previous page. By using the Serial.println() function in your program you can send data back to your computer, this greatly aids in debugging a program. Make sure that the speed on the serial port viewer is matched by the speed set in the program. The speed in the serial port viewer can be edited in the lower right corner, by default Arduino sets this to 9600, but the Yeti supports speeds up to 57600 Baud.





12.2. Wireless INFRA RED TRANSCEIVER

The Yeti comes standard with just the wireless USB to serial adaptor, this way of uploading programs can be more flexible then the wired connection since there is no wire to get in the way. The wireless convertor does not support feedback through the serial port viewer at the moment, but it is possible for an experienced programmer to make this work, see appendix if you really want wireless serial communication in your program.

13. CALIBRATION

For Yeti to walk stably a correct and carefully executed calibration program is necessary, it only has to be executed the first time you start Yeti, or after modifications to its body, legs or feet. A software calibration is necessary so the ATmega processor knows what the end points and center points are for each servo. These end points vary based on assembly of the Yeti. From the factory, Yeti does have base end points in its memory, but considering how much these points can fluctuate, we highly recommend following the calibration procedure

13.1. Hardware calibration

Before starting the software part of the calibration, it is very important that the servos are centered. Once the calibration program has been loaded into the Yeti, it will automatically center each servo, so you should unscrew the servo arms from the servos and remove them. After the program has centered the servos, you can reassemble them with the knowledge they are now centered. This goes for both the front and bottom servo. After the servos have been centered, you can continue to the software part of the calibration.

13.2. Software calibration

To calibrate the Yeti, we have included a program which will guide you through the setup process of Yeti. When you complete the calibration program, Yeti will automatically store the values you selected in its EEPROM memory. EEPROM is a type of memory which will keep the stored values even if power is lost, since Yeti isn't always one, that's a handy feature.
To start the calibration procedure; in the Arduino IDE, navigate to the example folder->Yeti->Examples->Yeti_Calibration. Upload this program to your Yeti, either wired or wireless. The calibration process itself is identical for the two, but it is important to keep the wireless USB module aimed at the back of Yeti if you are using the wireless module. If communication is lost while Yeti was transmitting, you should reboot Yeti. After opening the Serial port viewer, you should see the following being printed once Yeti has started: "Yeti calibration procedure started Are you sure you want to calibrate me? Type Y or N and press enter"

If you are not seeing this, either Yeti still has to start, the upload has failed (you can just re-upload the program) or the wireless module is not aimed properly at the back of Yeti or has too much interference.
Type Y to enter the calibration routine, the first point to calibrate is having the body lean right.

The key here is that the left leg is just slightly above the ground before pressing “S”. Do not have Yeti leaning very far to the right, or it will fall over during walking, the left leg should be just slightly above the ground. Type + or – and press enter to edit the position, you have to type a lot of + and –‘s because Yeti is quite sensitive. To get it to lean even slightly, the amount of –‘s you have to enter can be as high as 20/30. Type S and press enter when you are finished.

After the point of lean right we go back to center, this is for verification. Type + and press enter again about 20/30 times or until Yeti is level. Type S and press enter when Yeti is level.

We move on to lean left, the same applies here, make sure that the right leg is just slightly suspended in the air, but Yeti should not be leaning very far to the left. Again type S and press enter when finished.

Yeti will now level itself again, and move on to the point of right leg forward. The back of the right leg should be just aft of the center of the left leg.

See the picture for clarification, it is quite important for the sake of stability that the right leg is not too far forward. When you are done, type S and press enter.

We move on to center, again for verification purpose. Type + or – as long as it takes to reach center, type S and press enter when done.

Last point to calibrate, left forward, the same applies here as it did for right forward;
make sure the leg is not too far forward for the sake of stability when walking.
Type S and press enter when you are done.

When the Serial port viewer prints this: “Left forward saved, we are done now” the calibration is complete and the program has saved the endpoint values to its memory. You can now upload a program of your choosing. If you uploaded a program where Yeti should walk, but it keeps falling over, redo the calibration procedure, and make the endpoints of the legs smaller, that will aid stability while walking.

14. 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 ouverture. 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.

Writing programs

Programmers are used to name the developing proces ‘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 texteditor, called ‘ARDUINO’. To write a YETI-program you might even use any other editor, such as Notepad or Microsoft Word, but we strongly advise to use ARDUINO IDE 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, a compiler is built into the Arduino IDE. To just compile a program without uploading it you can press the compile button, the Arduino IDE will automatically do all the hard work for you.

‘Uploading’ a program

Imagine the compiler has just completed the translation of a ‘C++’ program into a file with microcontroller instructions called ‘test.hex’. As a final step you may start the upload button in the ARDUINO IDE to transfer the output file by infrared communication to the YETI micro-controller.

This final step, in which the ARDUINO IDE program transfers the output file to the YETI, is called ‘Uploading’ a program. If you click upload in the Arduino environment the Arduino IDE will compile the program automatically, so using the compile button is only useful to check your program for mistakes.

Basic structure for a ‘C++’ Arduino-program

Basically an Arduino-program minimally requires the following structure:

```
void setup()
{

}

void loop()
{

}
```

‘void’ indicating ‘no return value’

As a general rule each ‘C’-instruction has to be terminated by a semi-colon, except a block terminator (terminating bracket).
The Arduino IDE will automatically display the source code in predefined colors according to some special categories. Syntax or specific ‘C++’-functions will be colored bright orange, object names are in bold dark orange and comments will be written in grey.

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

```
void setup/*intermediate text line*/()
{ /*any text lines*/
//one line of text
/* Multiple
text
lines
*/
//any text
/*
some more text lines
*/
}
```

‘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.

YETI will switch on its right 'eye'

```
#include <Servo.h>
#include <Wire.h>
#include <yeti.h>              //include the yeti library

yeti robot;                    //create a yeti object

int rightLED = 8;              //pin connected to the right led


void setup()
{
  robot.initYeti();            //initialize the yeti program
  pinMode(rightLED, OUTPUT);     //set the pin to output
  digitalWrite(rightLED, HIGH);  //switch the led on
}

void loop()
{

}
```

#include "yeti.h", "Wire.h" and "Servo.h"
At the point where this is in the program, the compiler will insert the file defined, in this case 3 different files are inserted. yeti.h includes all of the functions to make the yeti work, the other two are used to make the yeti.h program work.

yeti robot;
This creates an object of yeti called robot, all of the functions for yeti can be called by placing robot. in front of the function. The creating of the robot object also sets up part of the microprocessor for duty as Yeti's brain.

int rightLED = 8;
This creates a variable for the right led, it is connected to digital pin 8 on the microprocessor. The left led is connected to digital pin 2 on the processor.

robot.initYeti();
This sets up the rest of yeti's basic functionalities that can't be started with just the "yeti robot;" command (starting of the servo's for example).

pinMode(rightLED, OUTPUT);
This sets digital pin 8 as an output so it can drive things like the attached LED, without this the pin is configured as an input by default, which is of course inconvenient if you want to turn on a LED.

digitalWrite(rightLED, HIGH);
Sets the output voltage of digital pin 8 to 5V, LOW corresponds to 0V. By making this pin high we turn on the LED, which is connected to this pin.

The "void loop()" is empty in this example since the processor doesn't have to do things repetitively. All it has to do is turn on a led once, so there is no point doing that thousands of times a second.

Another example:

```
#include <Servo.h>
#include <Wire.h>
#include <yeti.h>          //include the yeti library

yeti robot;                //create a yeti object
int rightLED = 8;          //pin connected to the right led

void setup()
{
  robot.initYeti();                    //initialize the yeti program
  pinMode(rightLED, OUTPUT);           //set the pin to output
  digitalWrite(rightLED, HIGH);        //switch the led on
  robot.beep(5000, 1000);              //make the beeper beep for 1
second
  delay(1000);
  robot.beep(2200, 1000);              //make the beeper sound for ano-
ther second
}

void loop()
{
}
```

robot.beep(5000, 1000);
This function from the yeti library makes the beeper which is on the Yeti's main control board sound at 5000 Hertz for 1 second, or 1000 milliseconds. This function is non-blocking, this means that you call it, it does what it should do and while that is continuing in the background, the program goes on. So we have to be careful if we want two different sounds right after each other, we have to slow the program down in between those two different sounds.

delay(1000);
In the last function we talked about non-blocking and how we have a problem if we want two different sounds right after each other. That is where the delay function comes in, it stops the program for the specified amount of time, in this case a 1000 milliseconds or 1 second. This function halts the program until the timeout has passed. Stuff that is running in the background is not stopped, so the beeper will continue to sound even if the delay function is called.

Make Yeti jump for joy

```
#include <Servo.h>
#include <Wire.h>
#include <yeti.h>                    //include the yeti library

yeti robot;                  //create a yeti object

void setup()
{
  robot.initYeti();                   //initialize the yeti program

  robot.moveForwardX(0);
  delay(300);
  robot.moveBody(3);            //3 stands for swing right
  delay(400);
  robot.moveBody(4);            //4 stands for swing left
  delay(150);
  robot.moveBody(5);            //5 stands for level
}

void loop()
{

}
```

robot.moveForwardX(0);
This function makes Yeti move forward, in this case 0 steps are chosen because we only want to center the servos before making Yeti move from side to side. If you chose another value, for example 2, Yeti will make 2 steps forward, stop and then level itself.

Robot.moveBody(3);
This function makes Yeti move to one side or level itself(just the front servo! use robot.moveForwardX(0); if you want to center both servos). If you put a 3 between the brackets Yeti swings to the right, if you put a 4 instead of the 3 Yeti swings to the left. A 5 stands for centering the servo, or leveling. If you just want Yeti to walk, you should use the function “moveForwardX()”;.

Making Yeti walk!

This program will make Yeti take 10 steps forward.

```
#include <Servo.h>
#include <Wire.h>
#include <yeti.h>                 //include the yeti library

yeti robot;                       //create a yeti object


void setup()
{
  robot.initYeti();               //initialize the yeti program

  robot.moveForwardX(10);         //make Yeti walk 10 steps forward
}

void loop()
{

}
```

robot.moveForwardX(10);
This makes Yeti take 10 steps forward, when it has taken 10 steps it will right itself, so no need to call robot.moveForwardX(0); to do that.

15. YETI EXTENSION SETS

General overview

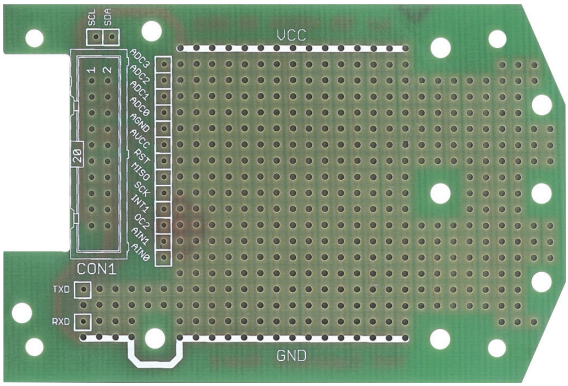
All extension sets will be connected to YETI's main PCB using a single 20-pole flat cable. The flat cable will also provide supply power to the sets and the I2C datatransfer to and from the extension sets.

15.1. YETI Experimental set YT-EXP1

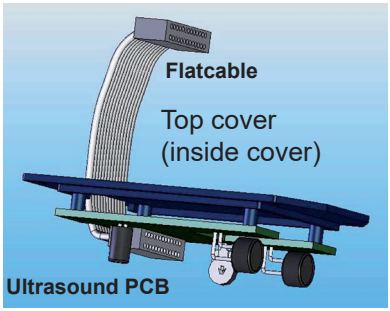
The experimental extension set has been designed to set up your own electronic designs and connect the experimental modules to the microcontroller.

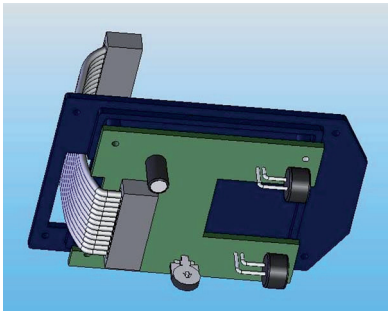
15.1.2. Partlist Experiment KIT YT-EXP1

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



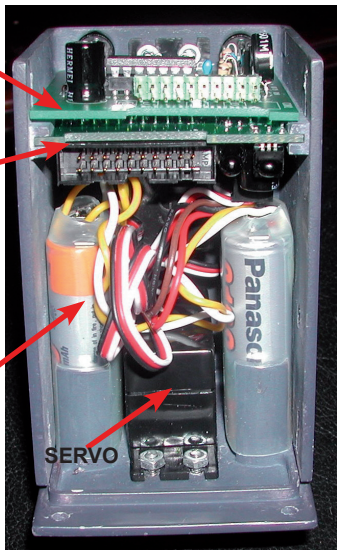
INSTALLING THE UPGRADE KITS





Installing Ultrasound PCB

Installing experiment PCB



Installing
Display PCB

15.2. 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.

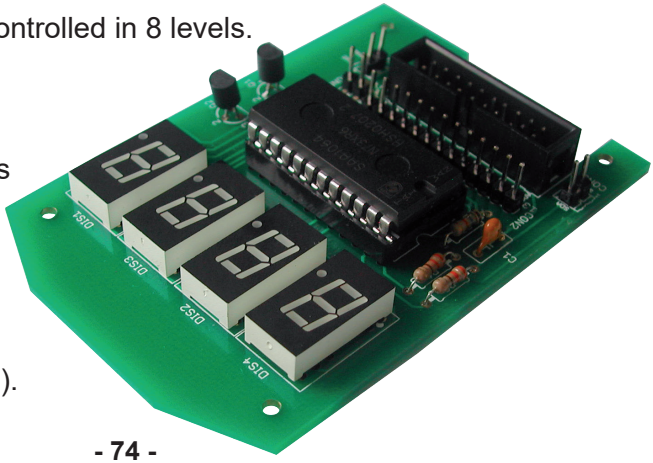
The YETI microcontroller controls the I2C driver chip. I2C is a standard communication protocol between electronic components, using only two signal wires SCL (serial clock) and SDA (serial data). We will need just one pair of a total of 20 wires provided by the flat cable.

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.

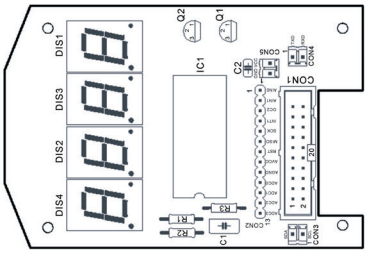
An alternative display method allows using two displays without multiplexing. These displays can be activated without switching on and off. To do so one display (Display 1) must be using P1-P8, whereas the other display (Display 3) will be using P9-P16. The software example suggests the display units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

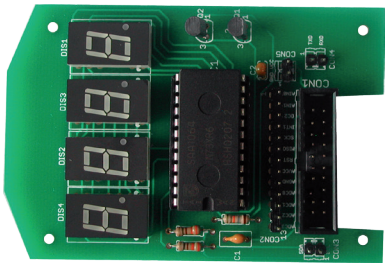
Display arrangement:
Display 4 Display 2 Display 3 Display 1

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.

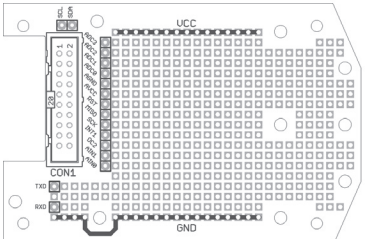
14.4. Parts list YETI Display Kit YT-DSP2

PCB-DSP	YETI DISPLAY PCB
R1	330R ***(Or, or, brn, gld)***
R2	330R ***(Or, or, brn, gld)***
R3	18K ***(Brn, grey, or, gld)***
C1	2,7nF ***(272)***
C2	100nF ***(104)***
Q1	BC547B/C ***(polarity!)***
Q2	BC547B/C ***(polarity!)***
D1	8-segment display common anode ***(polarity!)***
D2	8-segment display common anode ***(polarity!)***
D3	8-segment display common anode ***(polarity!)***
D4	8-segment display common anode ***(polarity!)***
IC1	SAA1064 ***(polarity!)***
S1	IC-socket, 24-pins, 600mil ***(polarity!)***
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 (2 pcs.)	Flat cable connector, 20-pins, female
F1	Flat cable, 20-wires, 10cm

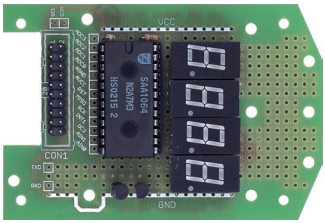


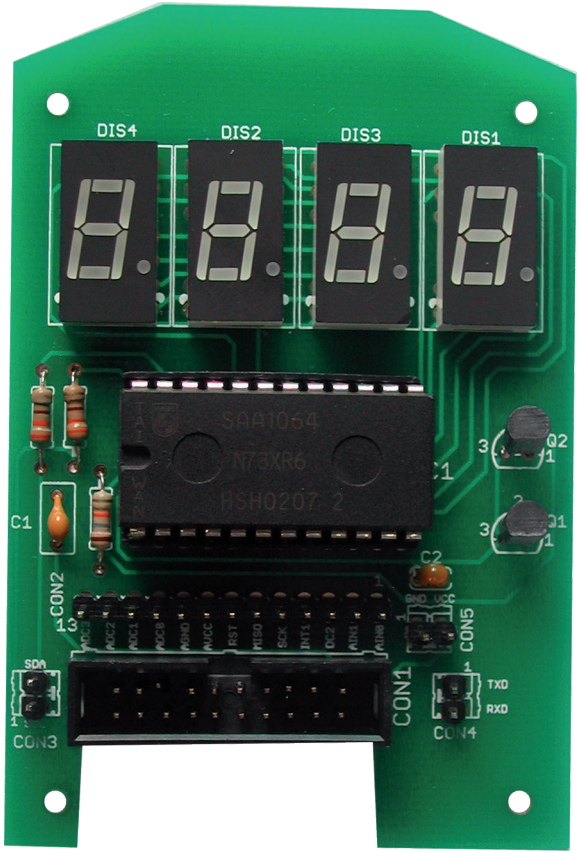


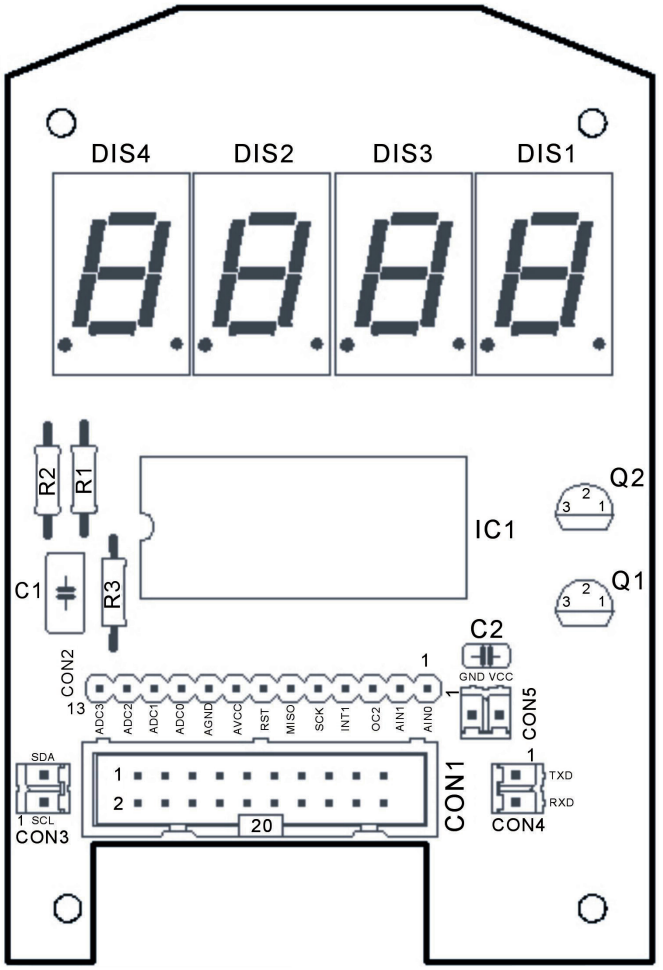
Display build on orginal Display PCB YT-DSP2 KIT



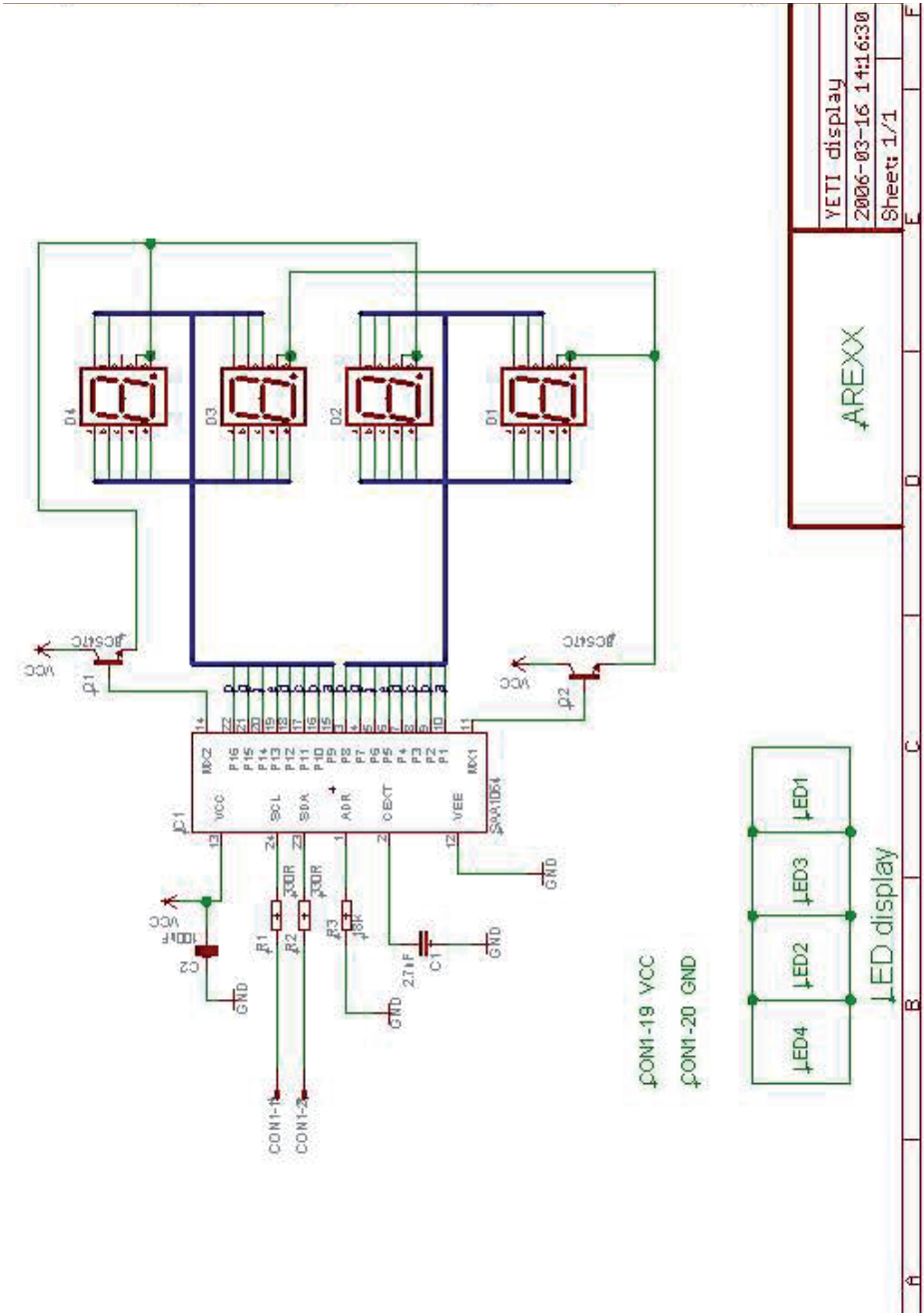
Display build on
Experiment PCB




DIS4
DIS2
DIS3
DIS1
SAA1064
N73XR6
R2
R1
C1
R3
IC1
Q2
Q1
C2
GND VCC
CON2
13
1
ADC3
ADC2
ADC1
ADC0
AGND
AVCC
RST
MISO
SCK
INT1
OC2
AIN1
AIN0
CON5
SDA
SCL
CON3
1
2
20
CON1
TXD
RXD
CON4


DIS4
DIS2
DIS3
DIS1
R2
R1
C1
R3
IC1
Q2
Q1
C2
GND VCC
CON2
13
1
ADC3
ADC2
ADC1
ADC0
AGND
AVCC
RST
MISO
SCK
INT1
OC2
AIN1
AIN0
CON5
SDA
SCL
CON3
1
2
20
CON1
TXD
RXD
CON4

15.5. Diagram Display set



15.6. 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.

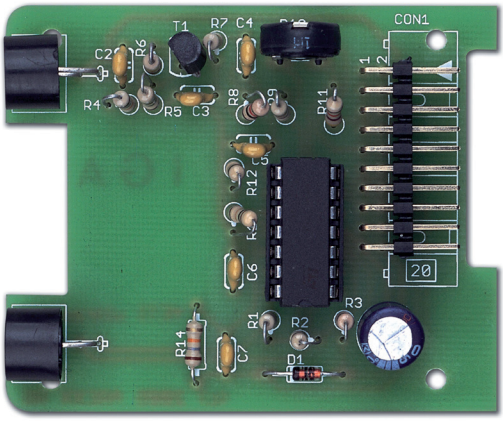
YETI instead uses a microphone for sound detection.

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.



15.7. 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 suply 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 signallevel 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 comparision level will result in high validation signal levels for fast response reflections and gradually lower validation signal levels for retarded response reflections.

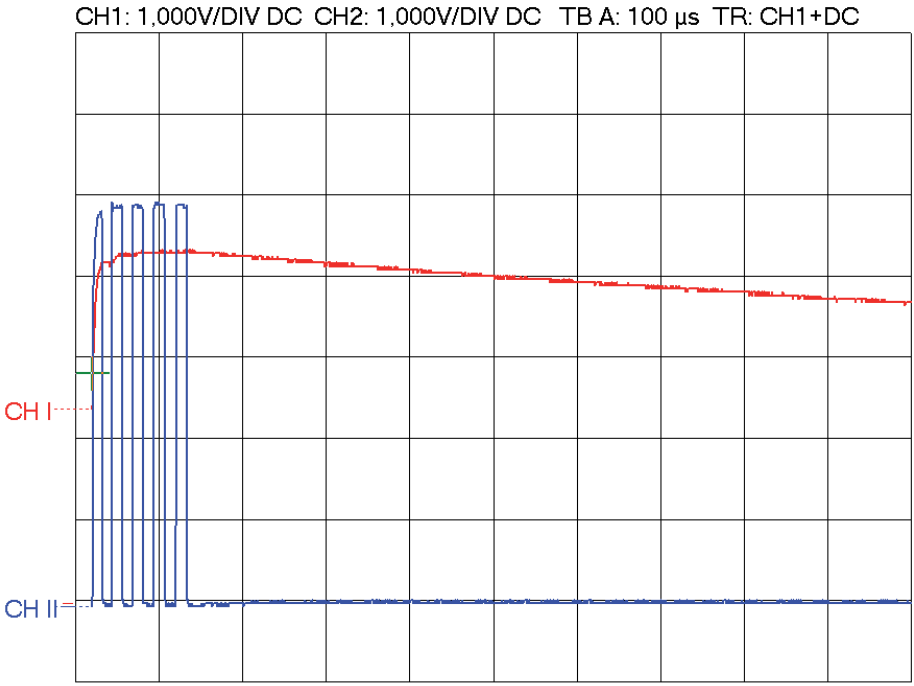
Of course the ultrasonic receiver system is extremely sensitive for any reflected signals, especially for signals from nearby objects in the transmitter’s vicinity.
In order to prevent reflections from the YETI’s head surrounding the ultrasonic transmitter and receiver, we will completely fill the robot’s head with cotton wool, including the volume between the band cable and the back of the head. See figure 1.

15.8. 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:

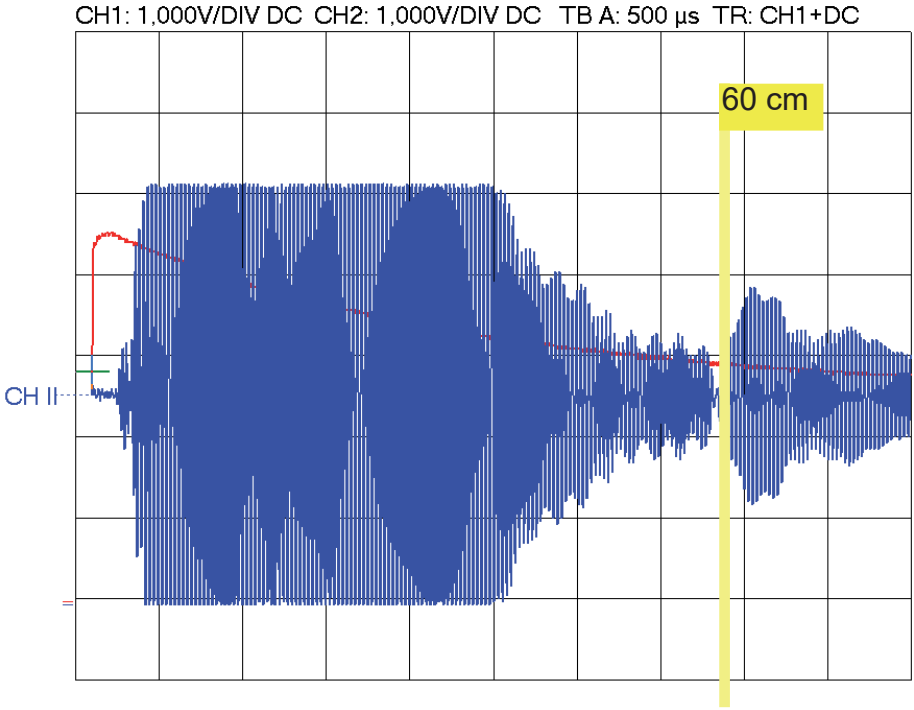


On this oscilloscope image you see two signals, measured on CON1-13 and CON1-15.

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).***

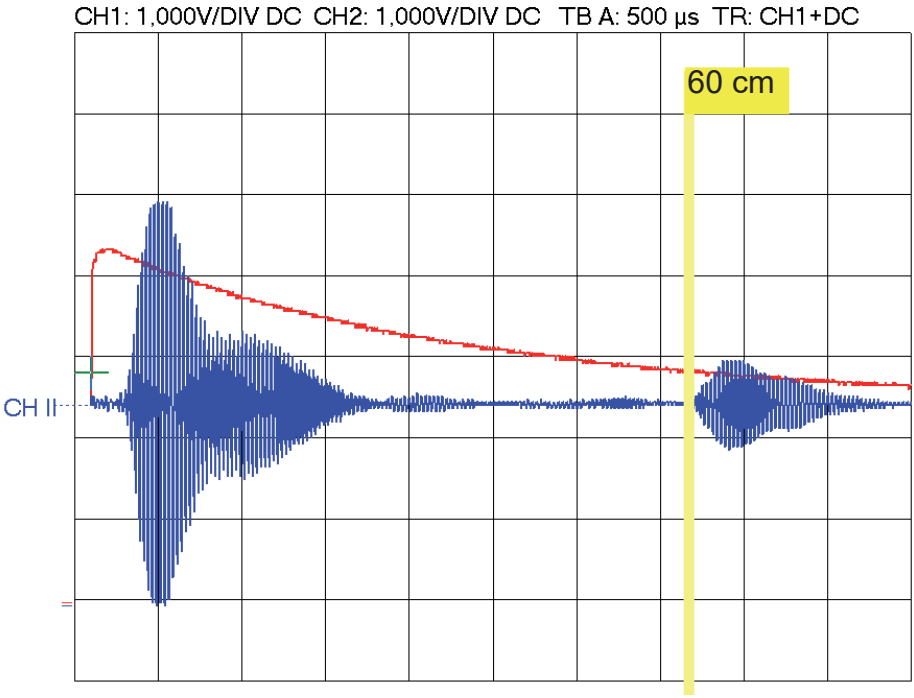
We mark the reflected signal at a distance of 60 cm. All strongly reflected signals are located above the reference line as a result of undesired direct reflections of YETI. These signals therefore cause an invalid measurement.

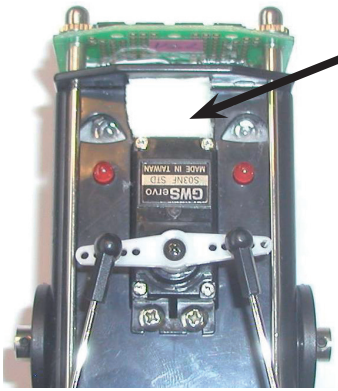
As you can see in our description the US receiver reacts very sensitive on all kind of reflected signals. This means it also reacts on false reflections inside YETI's head.

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)!

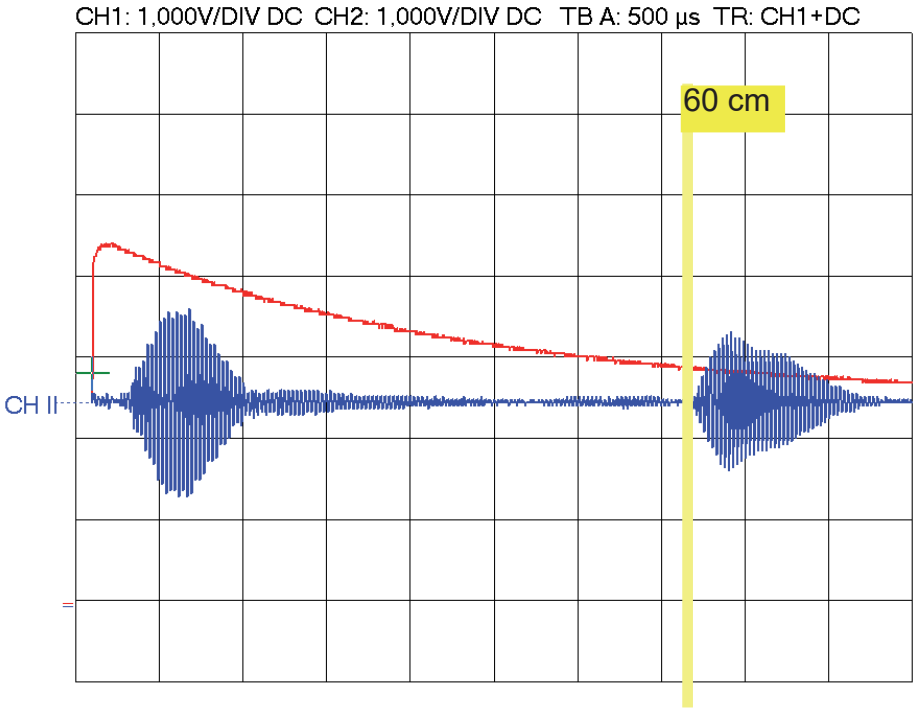
Step 1.
Yeti's head, completely filled with cotton wool.

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 measuments.





Step 2.
Filtering the outside reflections



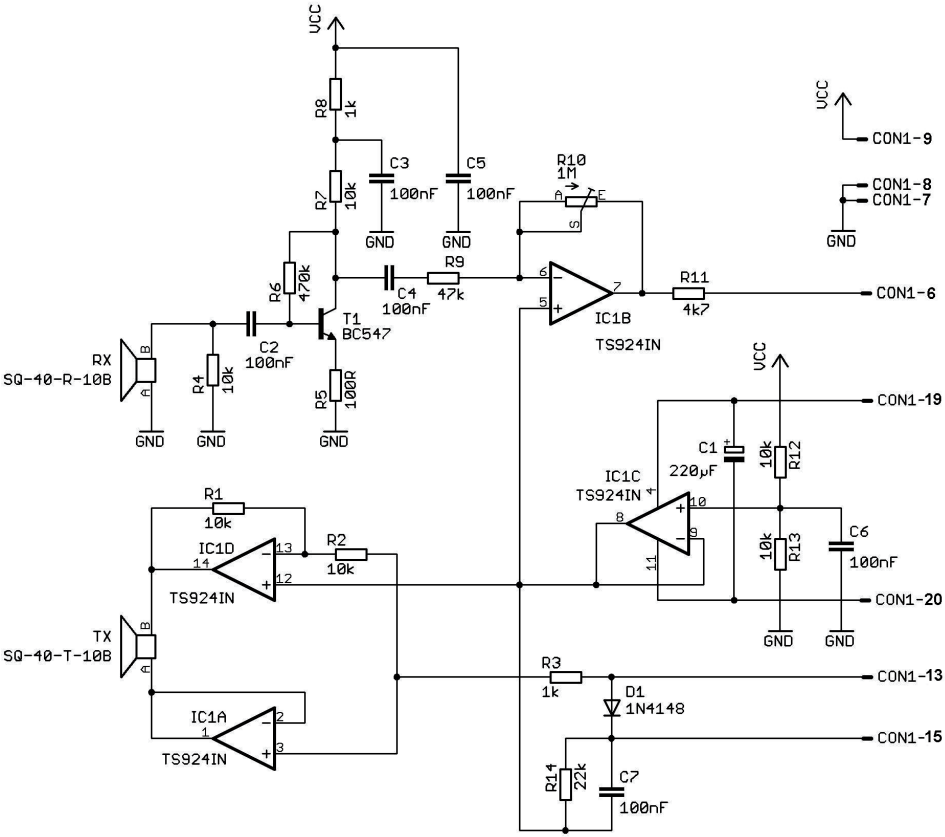
On the above image all reflected signals stay under the reference line. Now you can measure distances accurately. The time between 5 transmitted pulses and the reflected signal is calculated by the microprocessor and translated into an actual distance. This distance can be shown on a display.

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

15.9. Parts list YETI Ultrasonic set

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

15.10. Diagram Ultrasonic set



RX
TX
T1
R7
C4
R10
CON1
C2
R6
R4
R5
C3
R8
R9
R11
C5
R12
R13
C1
C6
R3
R1
R2
C1
R14
C7
D1
1
2
20

T1
R7
C4
CON1
C2
R6
R4
R5
C3
R8
R11
R12
C6
R3
R1
R2
R14
C7
D1
1
2
20

15.11. YETI programmer expansion board YT-PRG

The programmer expansion board has been designed to make it possible to program the Arduino Yeti with a wired programmer. While this might not seem like an upgrade, a wired connection has many advantages versus the wireless connection. The wired connection is up to 24 times faster than the wireless connection, the wireless connection is vulnerable to light and interference and the wired connection supports faster serial feedback from a program.

The programmer expansion board also has a lot of room for experimental use, there are 389 free pads for you to use as you like. You could for example connect experimental modules or add your own designs.

14.11.2 Parts list programmer expansion board YT-PRG

PCB-PRG	Yeti programmer expansion board
R1	10K
C1	100nF
CON1-PCB	PCB connector, male, 20 pins, for flatcable
CON2	Pin header 13 –pins PCB montage
CON4, 5,6	Pin header 2 –pins PCB montage
CON6	Pin header 10 –pins PCB montage
CON7	Pin header 7 –pins PCB montage
CON1-FC (2 pcs.)	Flatcable connector, 20-pins
F1	Flatcable, 20-ires, 10cm

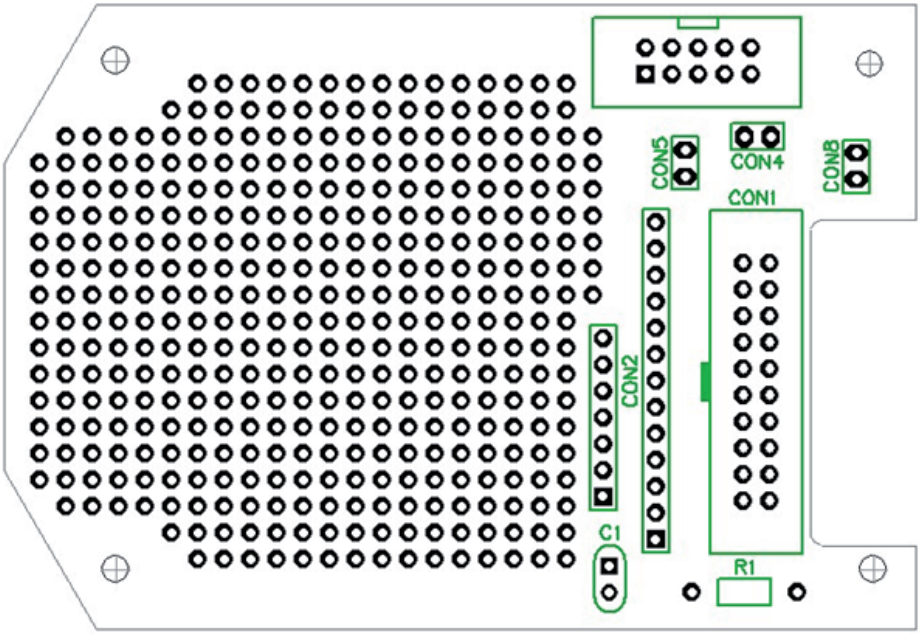
14.11.3 Use of the programmer board

The programmer expansion board can be used in two separate ways, if the display kit is also installed you can stack the programmer expansion board onto that board, the other way to use board is to use it as a standalone module.

If the display kit is installed on the your Yeti you will have to unsolder some connectors on that board (Connector 2, 4 and 5) and replace the small headers with longer headers. On the programmer expansion board you will have to solder the female connector 2, 4 and 5 to the bottom of the board so they can plug into the display kit. Take care when soldering the headers onto the bottom, make sure they are straight down or they won't align with the headers on the display kit. The other connectors should just be soldered to the top side of the board.

If the display kit is not installed, you should use the programmer expansion board as a standalone module. When used as a standalone module, you should not solder any headers to the bottom side of the board, because it will not clear the Yeti top plate when you try to install it. You should just use Connector 1 and the supplied flatcable.

Connector 8 can be bridge if you only want to use the wired high speed connection when the board is plugged in. If you want to select the upload speed while the board is plugged in, solder a header onto these pads and use a jumper to which between 2400 Bd and 57600 Bd upload speed.

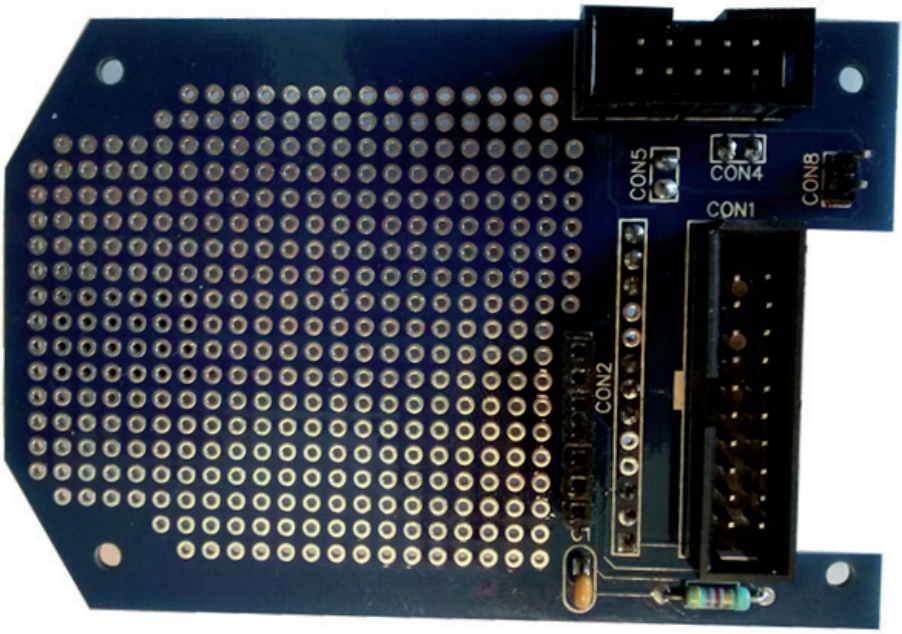


15.11.2 Parts list programmer expansion board YT-PRG

PCB-PRG	Yeti programmer expansion board
R1	10K
C1	100nF
CON1-PCB	PCB connector, male, 20 pins, for flatcable
CON2	Pin header 13 –pins PCB montage
CON4, 5,6	Pin header 2 –pins PCB montage
CON6	Pin header 10 –pins PCB montage
CON7	Pin header 7 –pins PCB montage
CON1-FC (2 pcs.)	Flatcable connector, 20-pins
F1	Flatcable, 20-ires, 10cm

14.11.3 Use of the programmer board

The programmer expansion board can be used in two separate ways, if the display kit is also installed you can stack the programmer expansion board onto



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

initYeti();
Initializer for the primary Yeti modules, this does not ready the microprocessor for the display or the ultrasonic modules!

wirelessSerialInit();
Initializer for the wireless serial module (USB to IR serial)

IRSerialprint(int TXData) / IRSerialprint(String TXData);
Prints either an integer or a string of characters through the IR Serial port to the computer or, in theory, another Yeti.

IRSerialprintln(int TXData) / IRSerialprintln(String TXData);
Does the same as the above, except that it prints a newline after the data has been send, this can be convenient if the Serial Port viewer on a computer is used. The difference between this and the above function is the “ln” at the end, this stands for newline.

IRSerialread();
Returns data from the Serial port if available, before calling this, check if data is available with Serial.available(); If this returns 0 then there is no serial data in the buffer to be read.

moveForwardX(int numberOfSteps);
Makes Yeti walk forward for the specified number of steps, this is the safest function to make Yeti walk forward.

moveForwardXNC();
The same as the above at its core, but with some small edits. This function does not center Yeti, making for smoother walking if this is called multiple times after each other. You also can not specify the amount of steps it should walk; it makes one step each time you call this function. This function has been made with the RF module in mind.

moveBackwardXNC();
The exact same as the above, but in the other direction; backwards instead of forwards.

turnRight(int angle);
Makes Yeti turn to the right, if angle is 0, it makes small steps to the right, if angle is anything but 0 it makes bigger steps.

turnLeft(int angle);
Makes Yeti turn to the left, if angle is 0, it makes small steps to the left, if angle is anything but 0 it makes bigger steps.

moveBody(int leanDirection);
Moves to the body to either the right or left side, or straight up. This function is useful if you want to make Yeti walk backwards for example, or if you want to make turns.
If you plan to use this function it is a good idea to define the following in the top of your program: 3 stands for lean to the right, 4 stands for lean left and 5 stands for straight up.

moveLegs(int walkDirection)
Moves the legs of Yeti into the specified position, options are: left forward, right forward and centered. This function is useful in conjunction to the previous function, to make Yeti turn for example.

If you plan to use this function it is a good idea to define the following in the top of your program: 6 stands for left forward, 7 for right forward and 8 for center.

beep(int frequency, int duration);
This function makes the buzzer beep at a set frequency, for a specified time period. The function is non-blocking, so the program keeps running after calling this, even if the duration hasn't expired yet. The frequency is in Hertz and the duration is in milliseconds, 1 second = 1000 milliseconds.

initDisplay();
Initializer for the Yeti display module, starts the display driver.

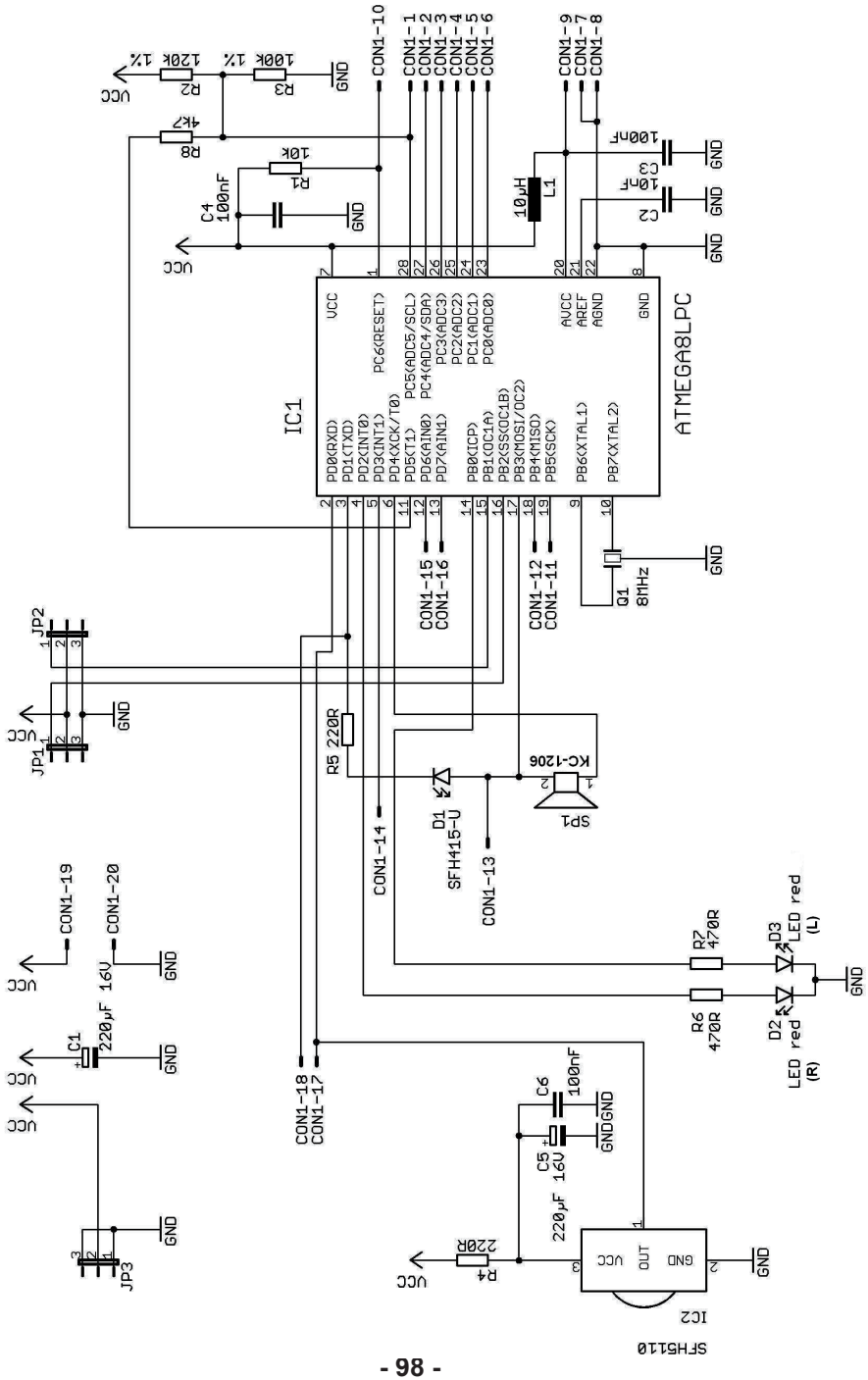
displayDigit(int displayNumber);
Displays one number from 0 to 9999, the not used places (for instance 1 has 3 unused digits) are filled with zero's.

displayDigit(int firstDigit, int secondDigit, int thirdDigit, int forthDigit);
Displays 4 small numbers, each from 0 to 9.

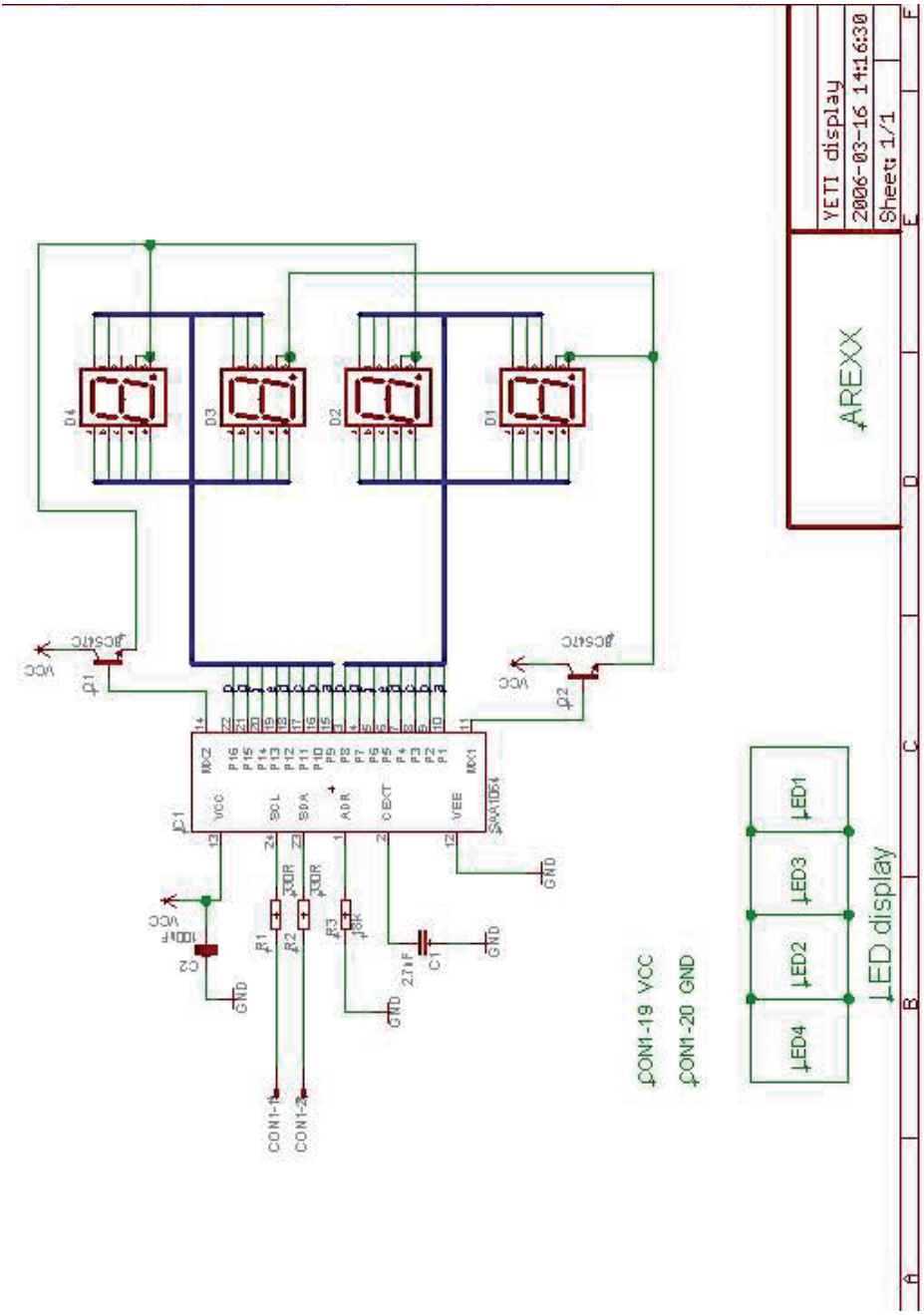
initPing();
Initializer for the Yeti ping module, sets up timers and such.

ping();
Requests the current value from the ping sensor. It first configures some timers, the buzzer might not function properly while the ping() function is busy. It sends 10 pulses, a faint click might be audible for young people. It returns the current distance to an object in centimeters, if there is no object detected it will return 0.

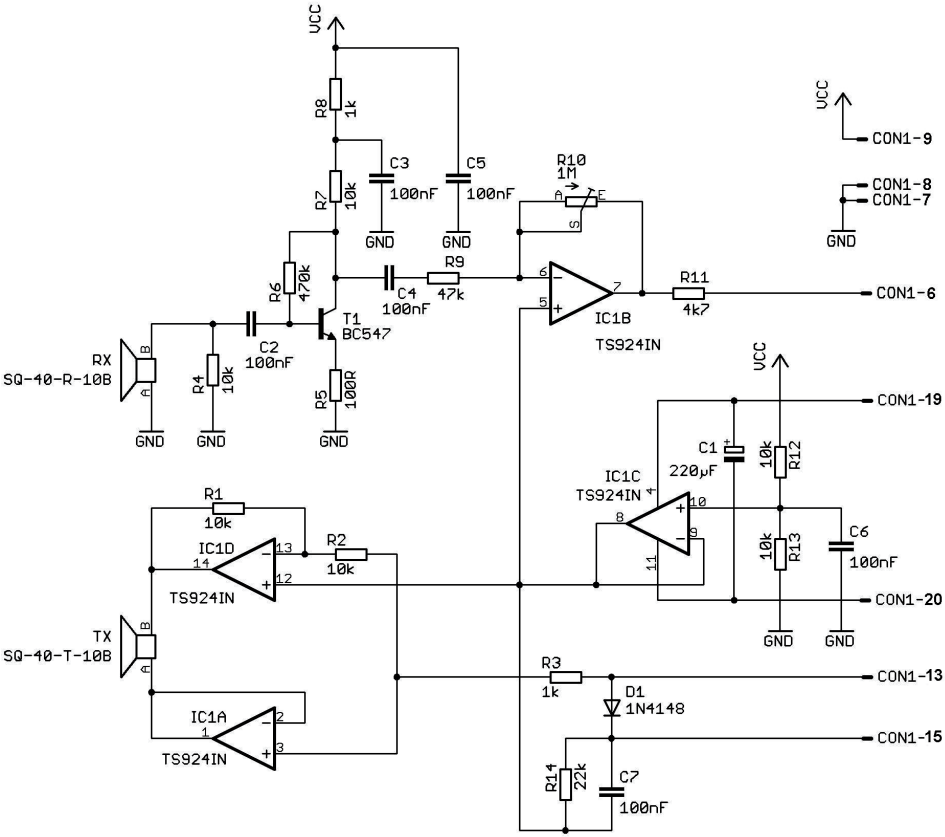
B. DIAGRAM YETI



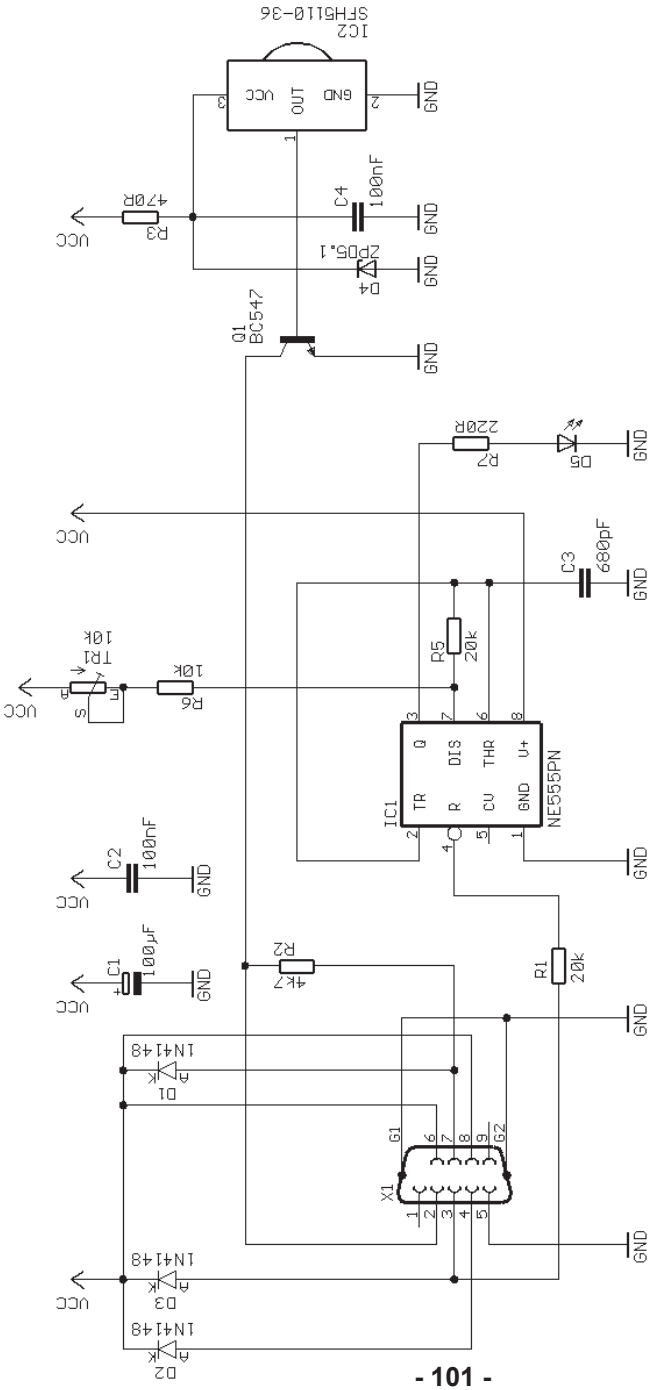
C. DIAGRAM DISPLAY MODULE



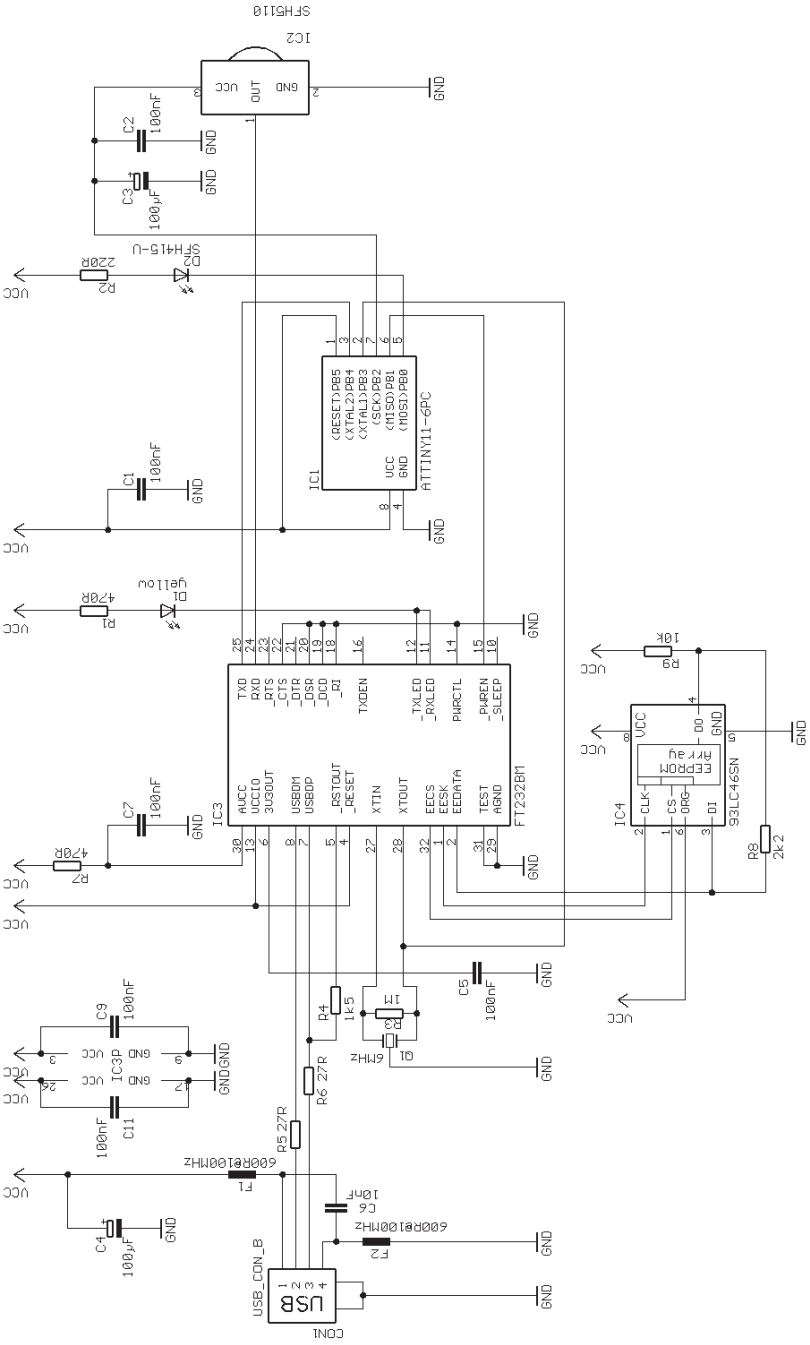
D. DIAGRAM ULTRASONIC MODULE



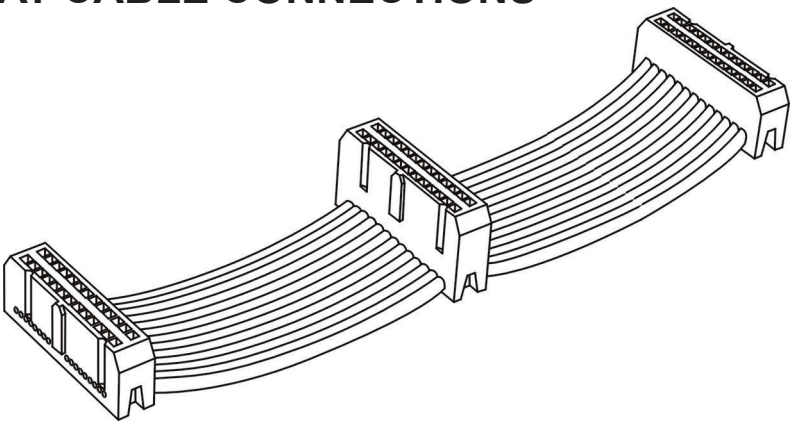
E. DIAGRAM RS-232 IR-TRANSCEIVER



F. DIAGRAM USB IR-TRANSCEIVER



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 ARDUINO IDE connection

- Check if you choose the correct com port
- Check if you choose the correct Arduino BOARD or processor

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.

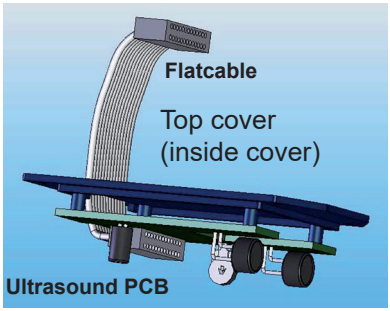
H.4.3. Things still do not work well

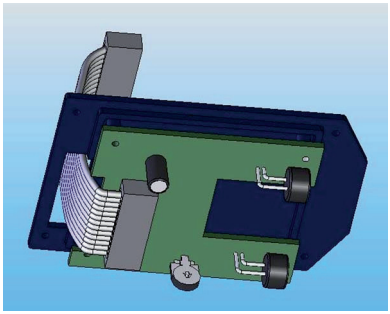
Check polarity of C8.

220µF/ at least 10V

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

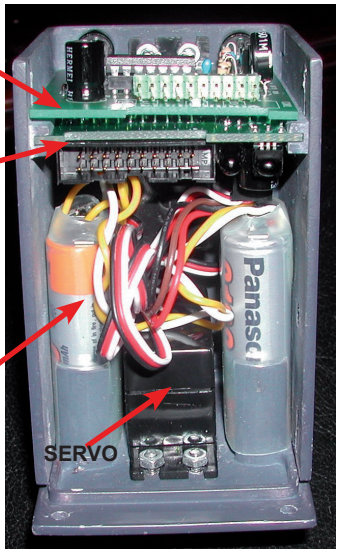
I. INSTALLING THE UPGRADE KITS





Installing Ultrasound PCB

Installing experiment PCB



Installing
Display PCB

Calculation table ADC-Values for Accu voltage

Author: Uwegw 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, f.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	
	5,5	2,50	945	
Accu full	5,4	2,45	927	In this range, all components should function without any problems.
	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	
Accu half full	4,6	2,09	790	In this range the voltage for the infrared receiver is too low.
	4,5	2,05	773	There is a possibility there will be some function failures.
	4,4	2,00	756	The Mega8 still has enough voltage.
	4,3	1,95	739	
Accu soon empty	4,2	1,91	721	In this range the Mega8 still functions. Other components could fail soon.
	4,1	1,86	704	The accu is almost empty and should be charged.
	4	1,82	687	
	3,9	1,77	670	
	3,8	1,73	653	
Accu empty	3,7	1,68	635	In this range, the minimum accu voltage of 0.9V per cell is achieved. This means that when
	3,6	1,64	618	you discharge the accu any further, it can be damaged.
	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