MyElectronicProjects Documentation

Release 0.0.0

ponty

December 28, 2011

CONTENTS

MyElectronicProjects

Date December 28, 2011

PDF MyElectronicProjects.pdf

CHAPTER ONE

ABOUT

Hobby electronic projects built by me.

Most of them are built on stripboard.

Links:

- home: https://github.com/ponty/MyElectronicProjects
- documentation: http://ponty.github.com/MyElectronicProjects

Design tool: EAGLE Light Edition

CHAPTER

TWO

STRIPBOARD DESIGN

Stripboard design representation in eagle:

- holes: copper should be cut or drilled here
- SMD: through-hole component, legs are drawn on top layer
- top layer: wires
- lines on documentation layer: wires
- bottom layer: original parallel strips of copper, only those are drawn, which are used for connection
- via: soldering points

Some components have no 3D view in the documentation.

CHAPTER

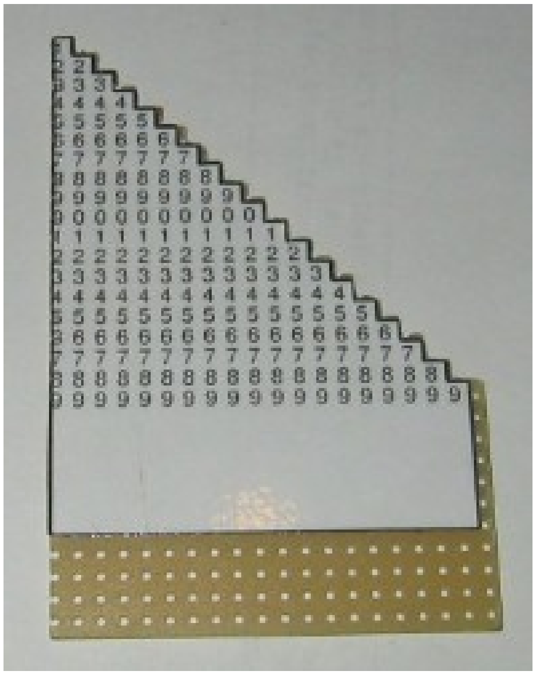
THREE

WIRE BENDING TOOL

Status: OK

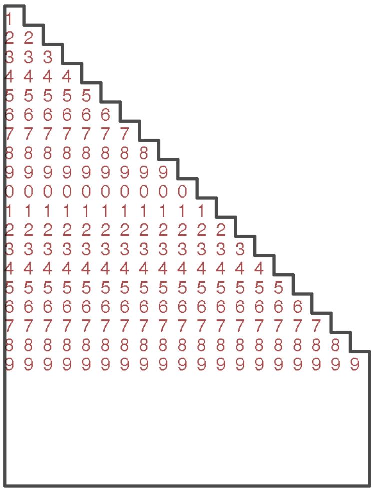
It is used for bending wires.

3.1 Image



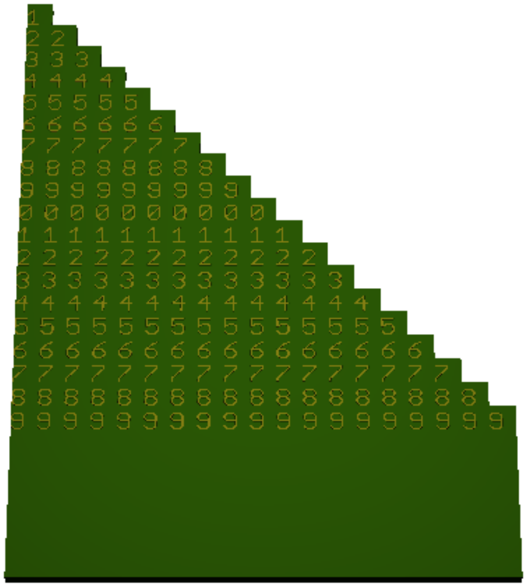
3.2 Board

Normal:



3.3 3D view

3.3.1 Front



3.3.2 Right side

CHAPTER

FOUR

DAPA AVR PROGRAMMER

Status: OK

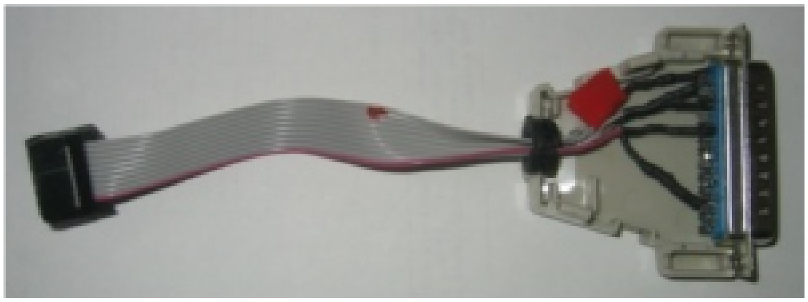
It is used for programming AVR controller and Arduino compatible boards using the parallel port.

4.1 Test on Ubuntu

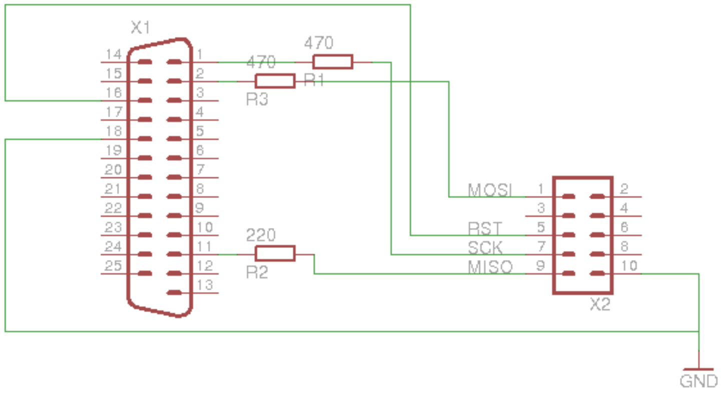
checking:

```
$ avrdude -patmega88 -cdapa

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e930a

avrdude: safemode: Fuses OK

avrdude done.  Thank you.
```

4.2 Image



4.3 Schematic



4.4 Partlist

Table 4.1:

part	value
R1	470
R2	220
R3	470
X1	
X2	

4.5 Sources

original design

Parallel port specification

ISP pinout

CHAPTER FIVE

FTDI CABLE

Status: OK

Special cable.

connections:

FTDI pin	signal	color	6p4c (RJ14) pin
1	gnd	red	4
2	cts		
3	5v	green	3
4	rxd	yellow	2
5	txd	black	5
6	rts		

standard color code is reversed

5.1 Sources

RJ14 pinout

RJ14 wiring details

CHAPTER

SIX

GARMIN ETREX DATA CABLE

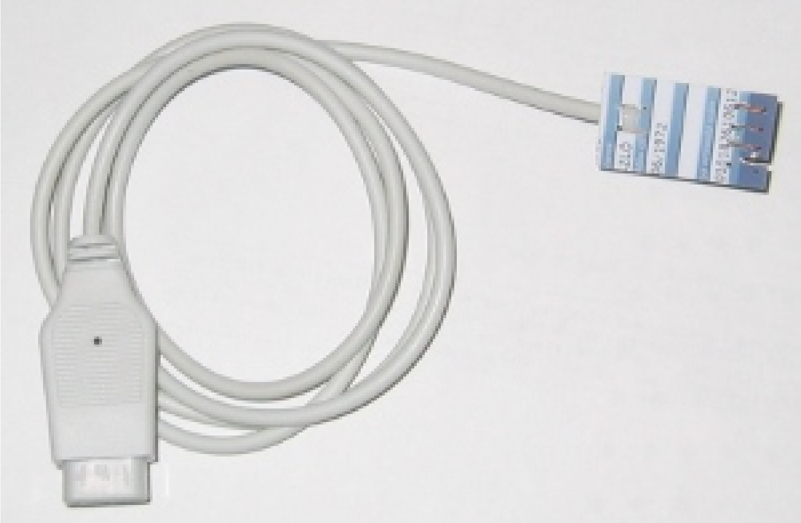
Status: OK

It is used for connecting Garmin eTrex to the serial port.

connections:

DB9 pin	garmin pin
3 (TxD)	2 (In)
2 (RxD)	3 (Out)
5 (GND)	4 (GND)

6.1 Images



6.2 Sources

original design

CHAPTER

SEVEN

SERIAL PORT LOOPBACK

Status: OK

It is used for testing the serial port.

Connected pins:

- 1-6-4
- 2-3
- 7-8

7.1 Images

7.2 Sources

original design

Serial port pinout

CHAPTER

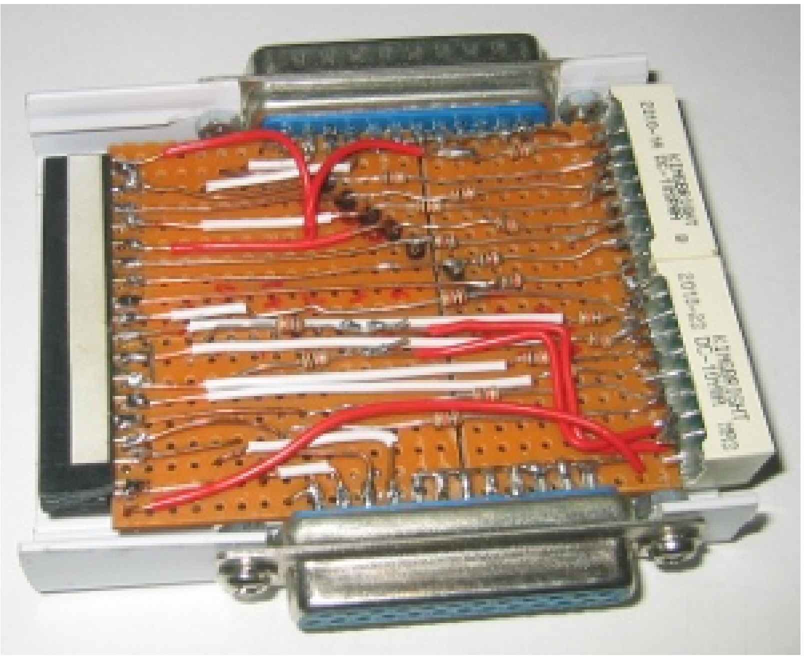
EIGHT

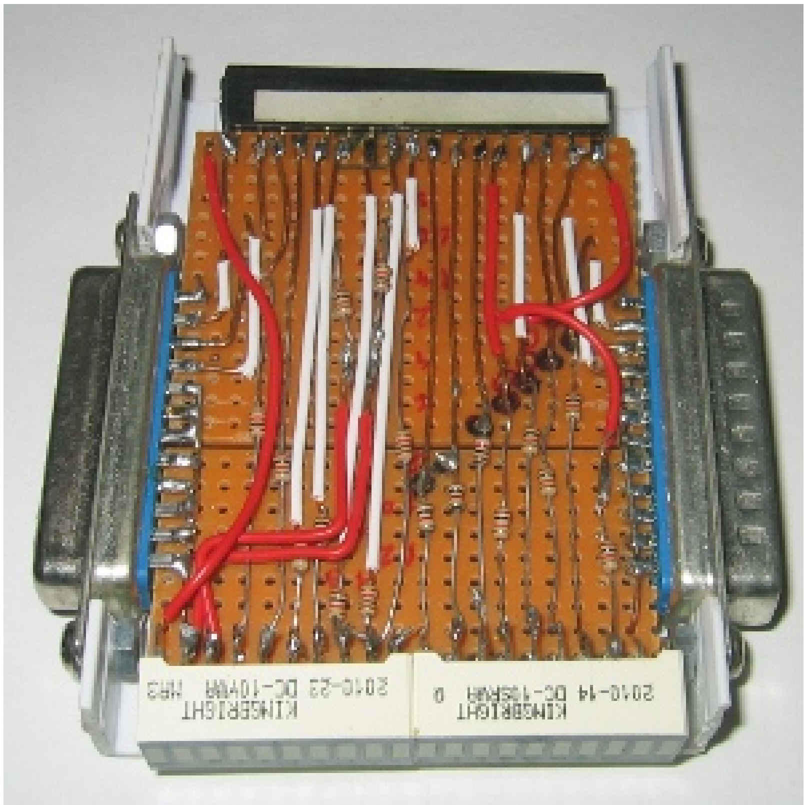
PARALLEL PORT MONITOR

Status: OK

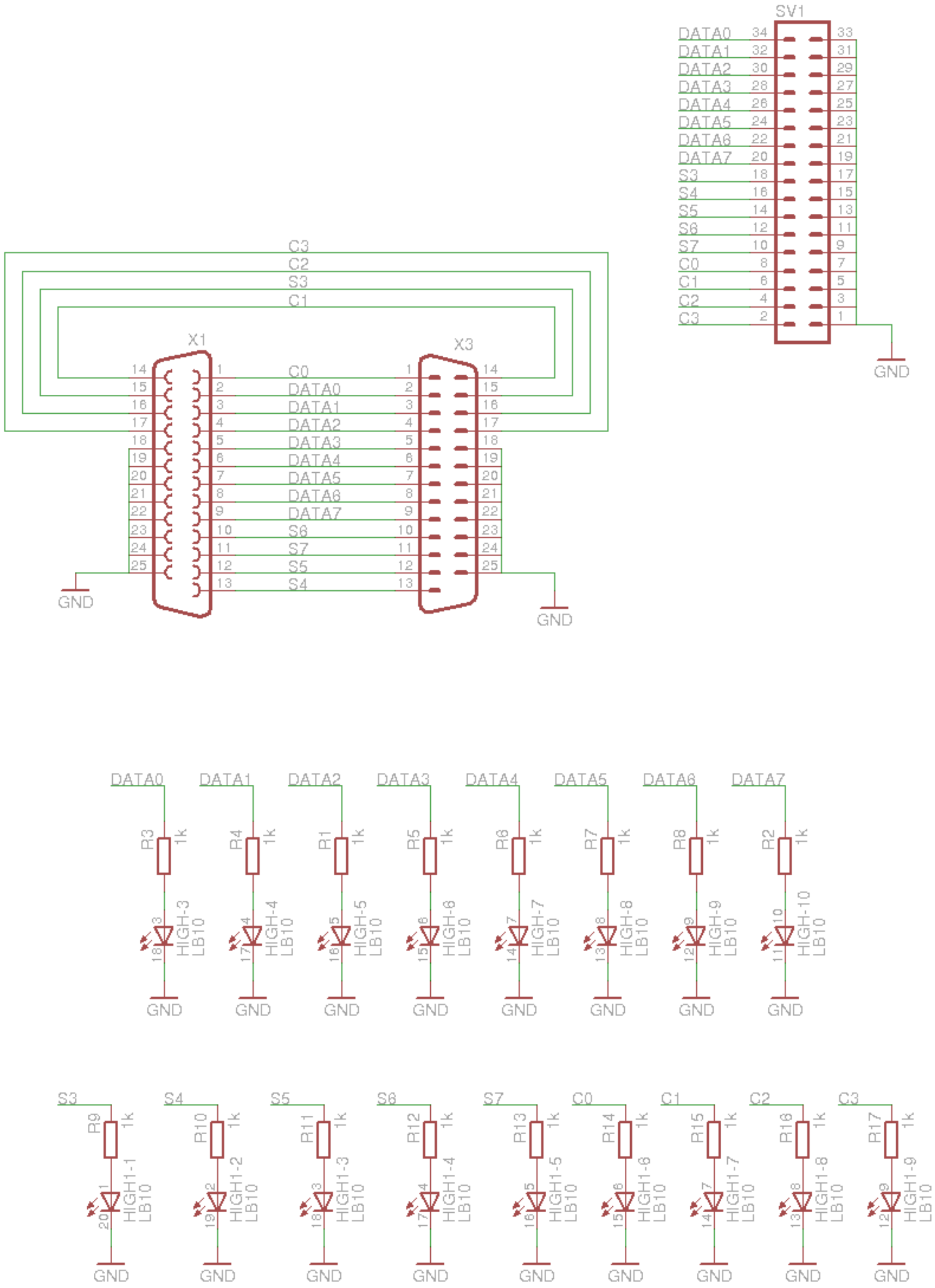
It is used for monitoring the parallel port signals.

8.1 Images



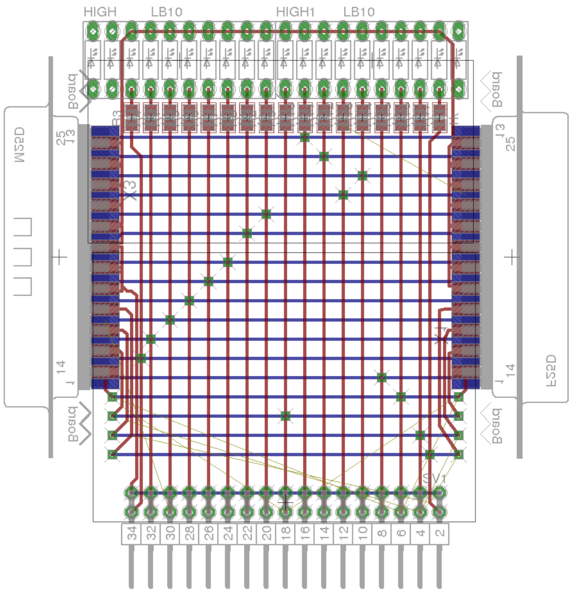

KINGBRIGHT
2010-23 DC-10YWA MR3
KINGBRIGHT
2010-14 DC-10SRWA 0

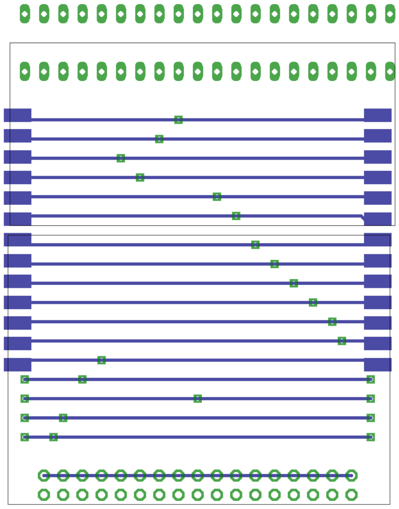
8.2 Schematic

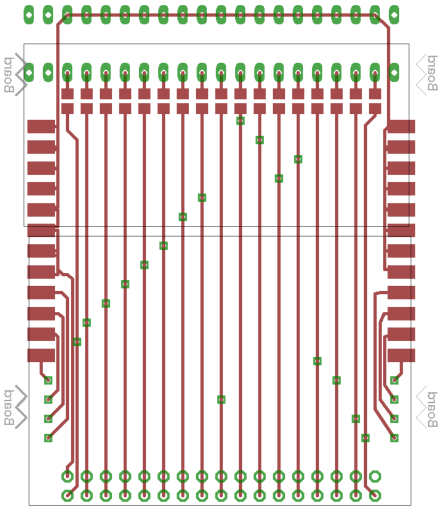


8.3 Board

Normal, bottom mirrored, wires only:


HIGH
LB10
HIGH1
LB10
25
13
14
1
34
32
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2





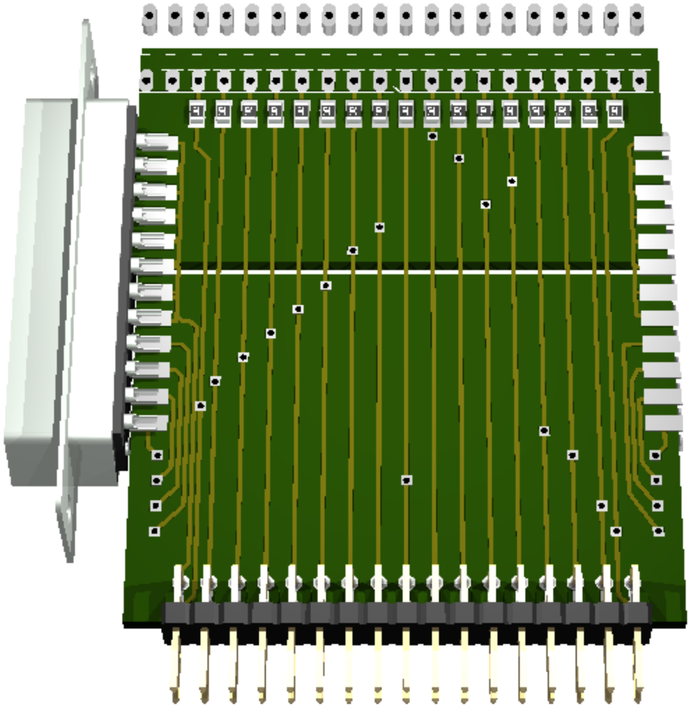
8.4 Partlist

Table 8.1:

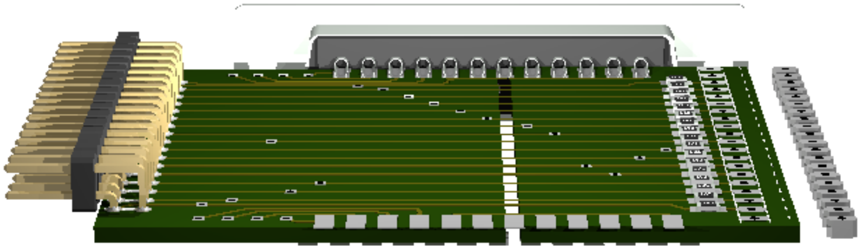
part	value	position
HIGH	LB10	(0.45 2.55)
HIGH1	LB10	(1.45 2.55)
R1	1k	(0.4 2.25)
R2	1k	(0.9 2.25)
R3	1k	(0.2 2.25)
R4	1k	(0.3 2.25)
R5	1k	(0.5 2.25)
R6	1k	(0.6 2.25)
R7	1k	(0.7 2.25)
R8	1k	(0.8 2.25)
R9	1k	(1 2.25)
R10	1k	(1.1 2.25)
R11	1k	(1.2 2.25)
R12	1k	(1.3 2.25)
R13	1k	(1.4 2.25)
R14	1k	(1.5 2.25)
R15	1k	(1.6 2.25)
R16	1k	(1.7 2.25)
R17	1k	(1.8 2.25)
SV1		(1 0.25)
X1		(2.175 1.525)
X3		(-0.175 1.525)

8.5 3D view

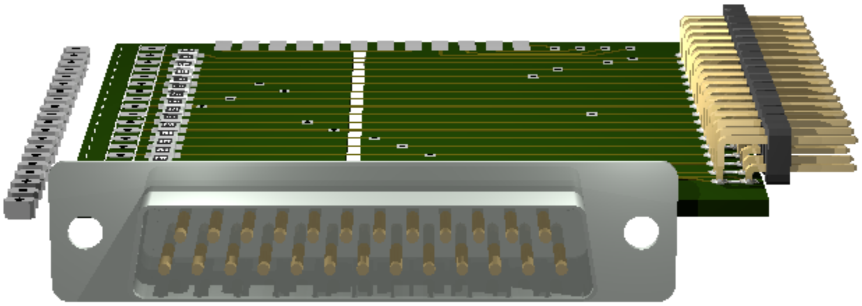
8.5.1 Front



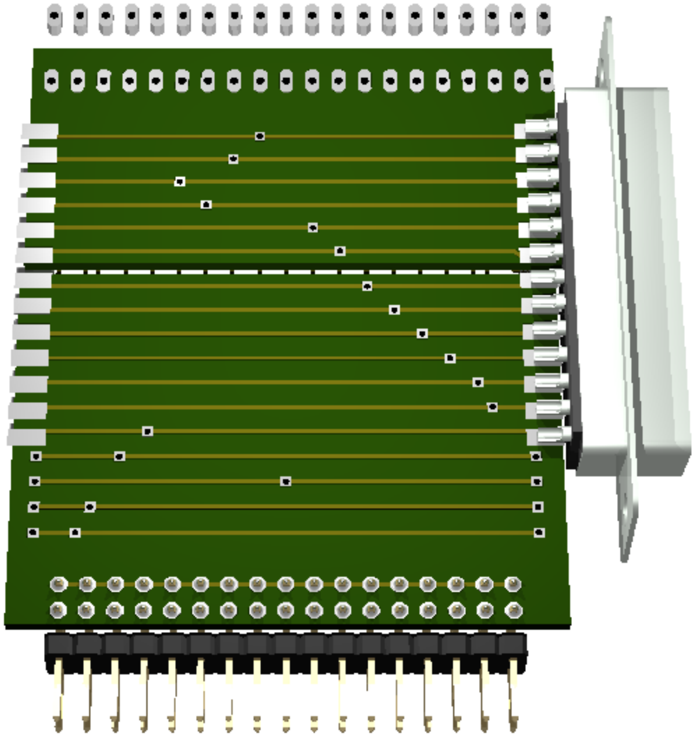
8.5.2 Right side



8.5.3 Left side



8.5.4 Bottom



8.6 Sources

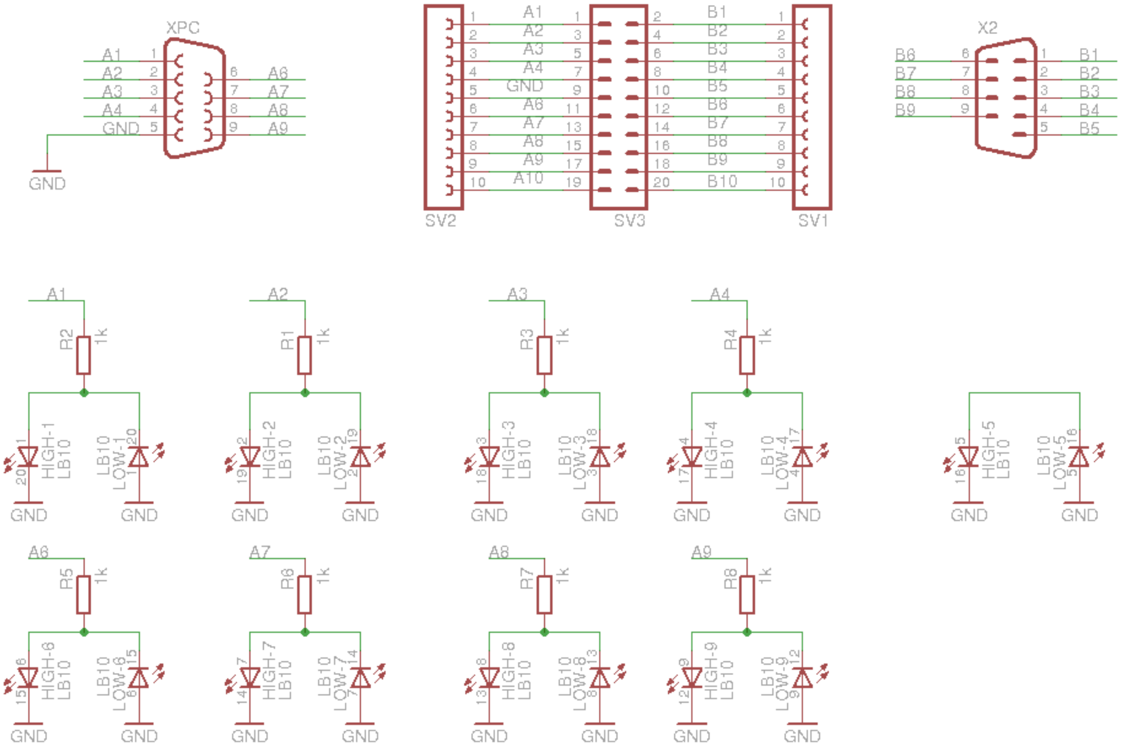
original idea

SERIAL PORT MONITOR

Status: OK

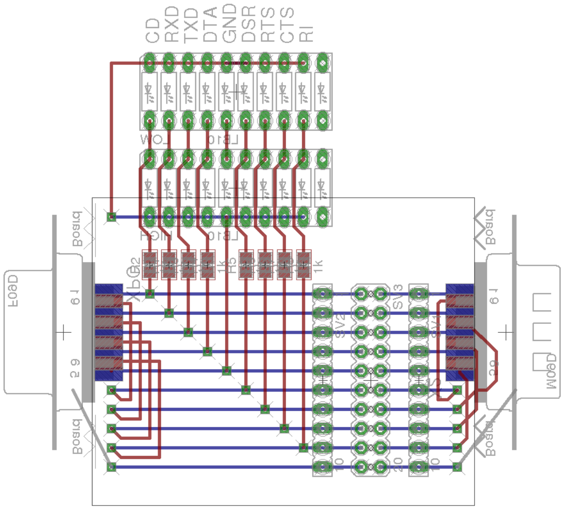
On each signal there is one LED for positive and one LED for negative voltage. It is easy to change connections or connect external parts. Examples: Loop-Back, Null Modem,..

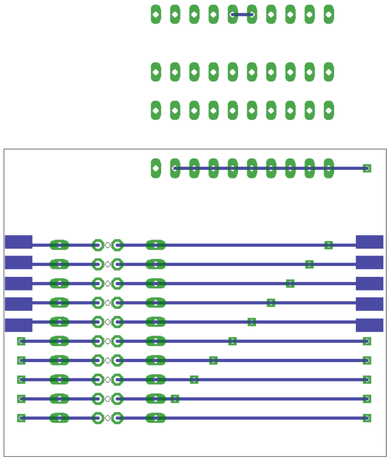
9.1 Schematic

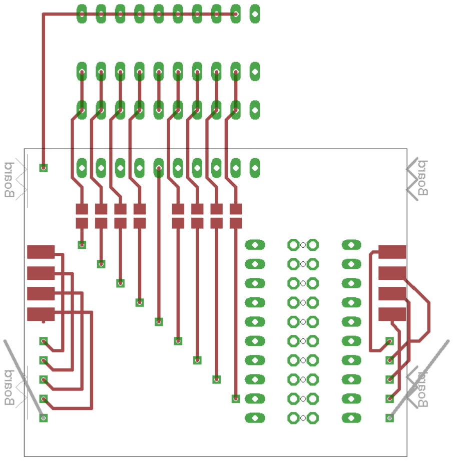


9.2 Board

Normal, bottom mirrored, wires only:


CD
RXD
TXD
DTA
GND
DSR
RTS
CTS
RI
Board
Board
Board
Board
SV2
SV3
1k




Board
Board
Board
Board

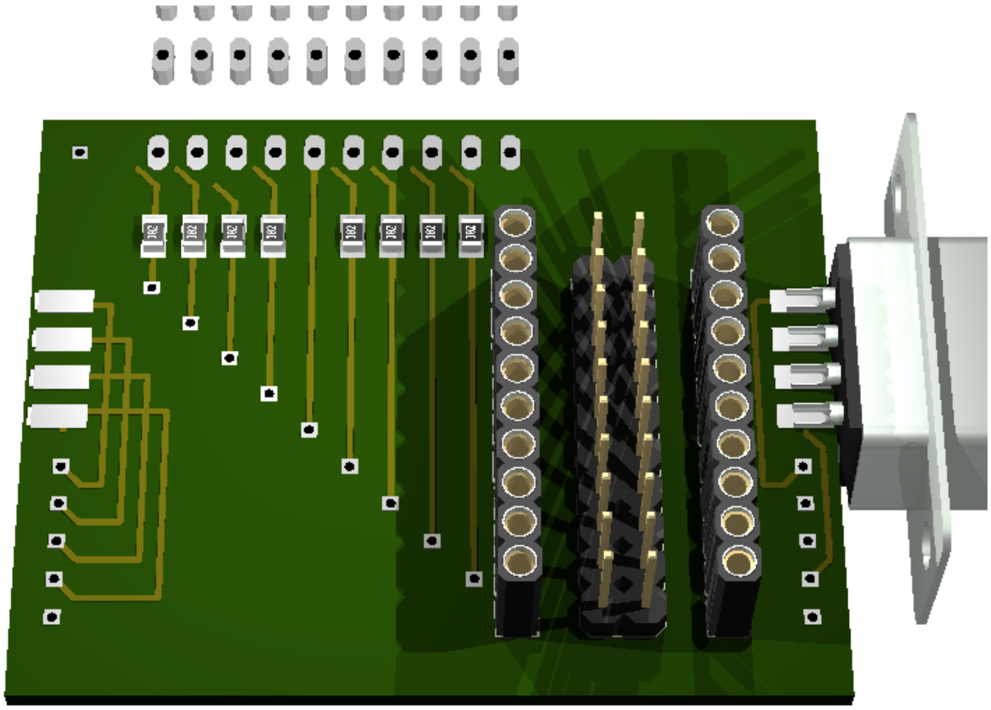
9.3 Partlist

Table 9.1:

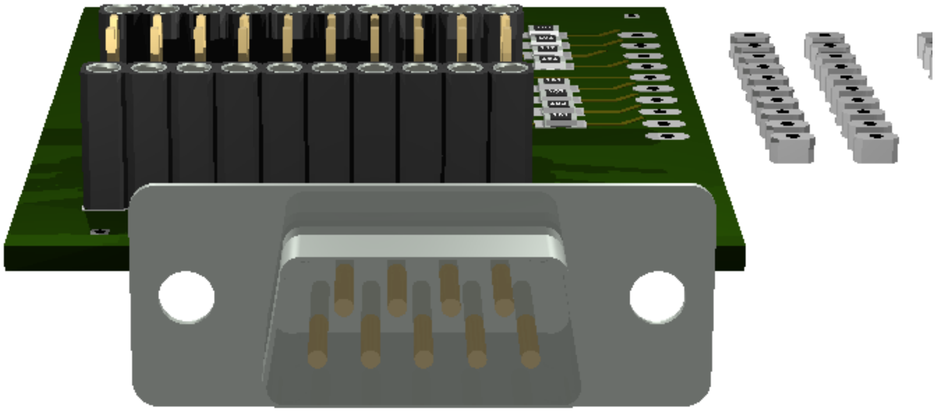
part	value	position
HIGH	LB10	(0.75 1.65)
LOW	LB10	(0.75 2.15)
R1	1k	(0.4 1.25)
R2	1k	(0.3 1.25)
R3	1k	(0.5 1.25)
R4	1k	(0.6 1.25)
R5	1k	(0.8 1.25)
R6	1k	(0.9 1.25)
R7	1k	(1 1.25)
R8	1k	(1.1 1.25)
SV1		(1.7 0.65)
SV2		(1.2 0.65)
SV3		(1.45 0.65)
X2		(2.15 0.9)
XPC		(-0.15 0.9)

9.4 3D view

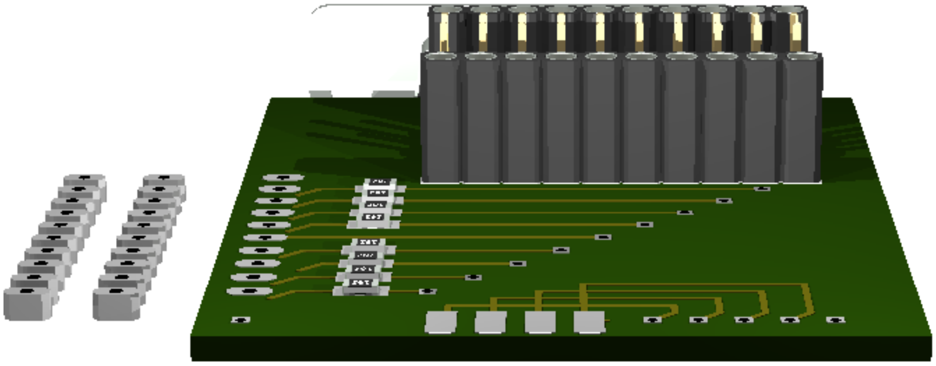
9.4.1 Front



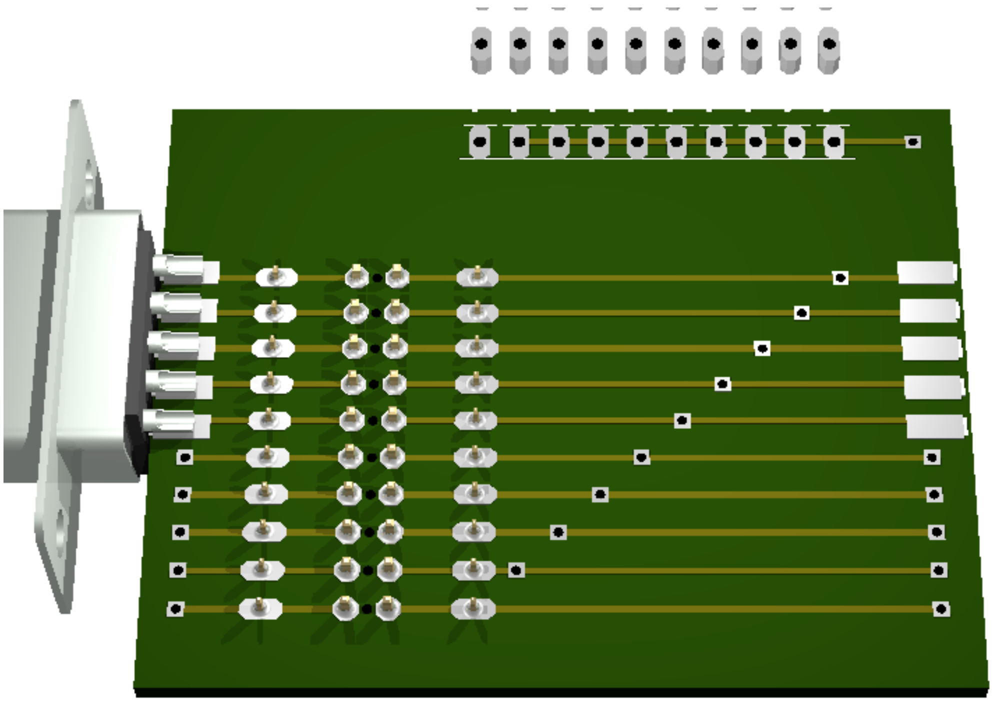
9.4.2 Right side



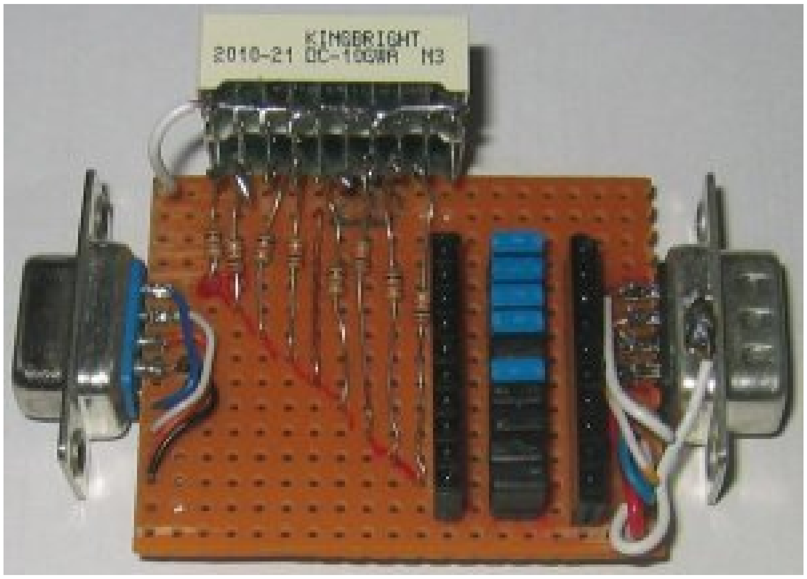
9.4.3 Left side

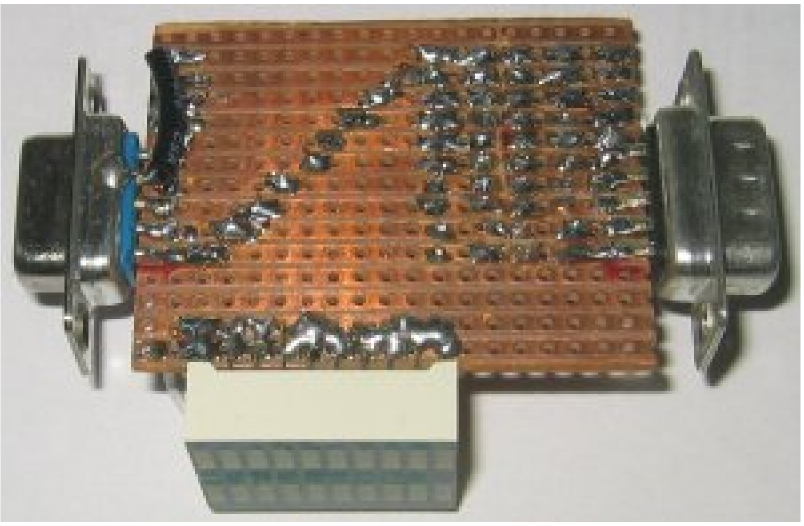


9.4.4 Bottom



9.5 Images





CHAPTER

TEN

STK200 AVR PROGRAMMER

Status: OK

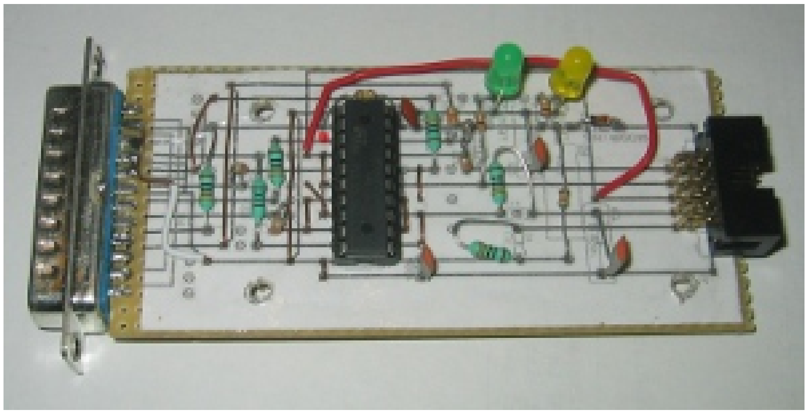
It is used for programming AVR controller and Arduino compatible boards using the parallel port.

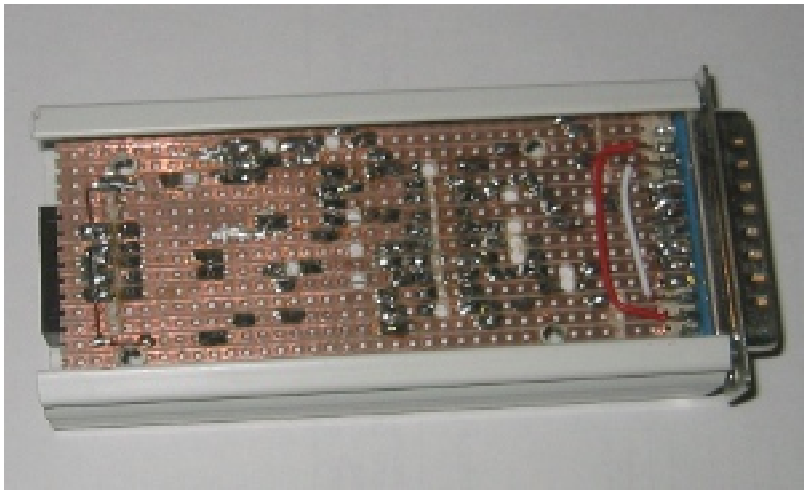
10.1 Test on Ubuntu

checking:

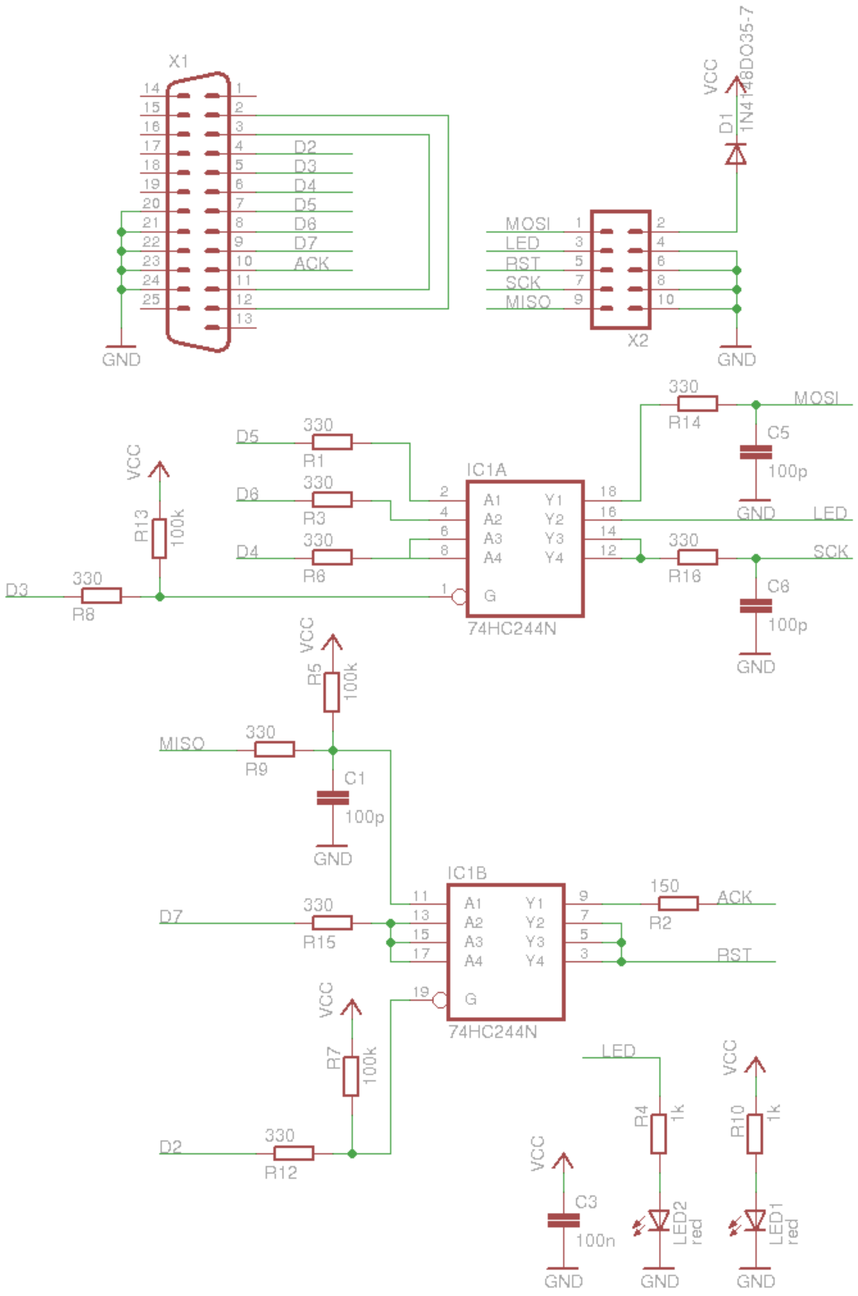
```
$ avrdude -patmega88 -cstk200

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e930a

avrdude: safemode: Fuses OK

avrdude done.  Thank you.
```

10.2 Image



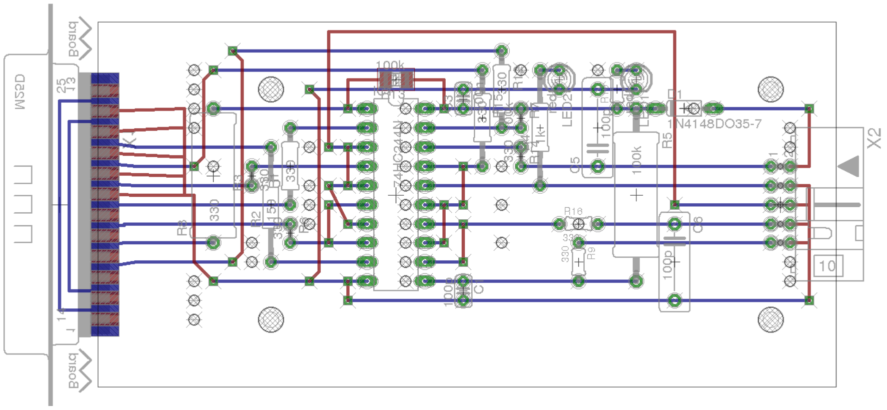


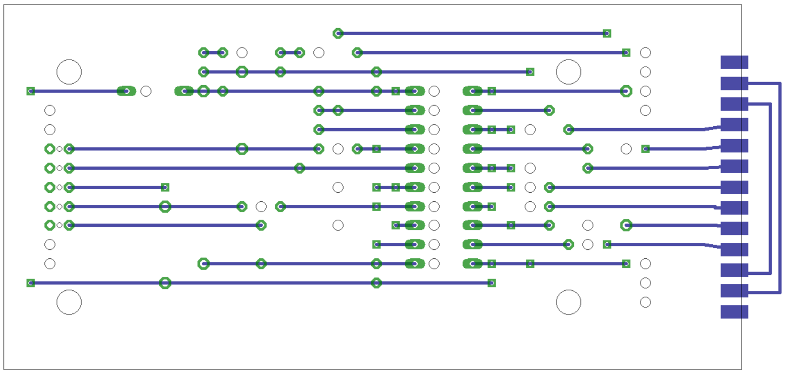
10.3 Schematic

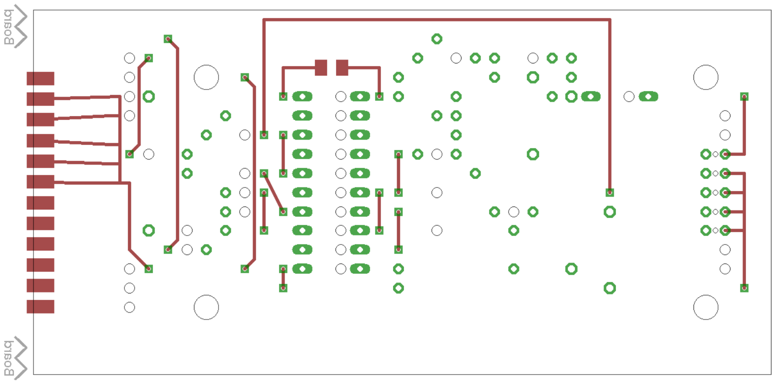


10.4 Board

Normal, bottom mirrored, wires only:


Board
25
13
100k
74HC244N
1N4148DO35-7
X2
10
R5
C5
100p
100k
330
LED2
R18
R9
C
Board




Board
Board

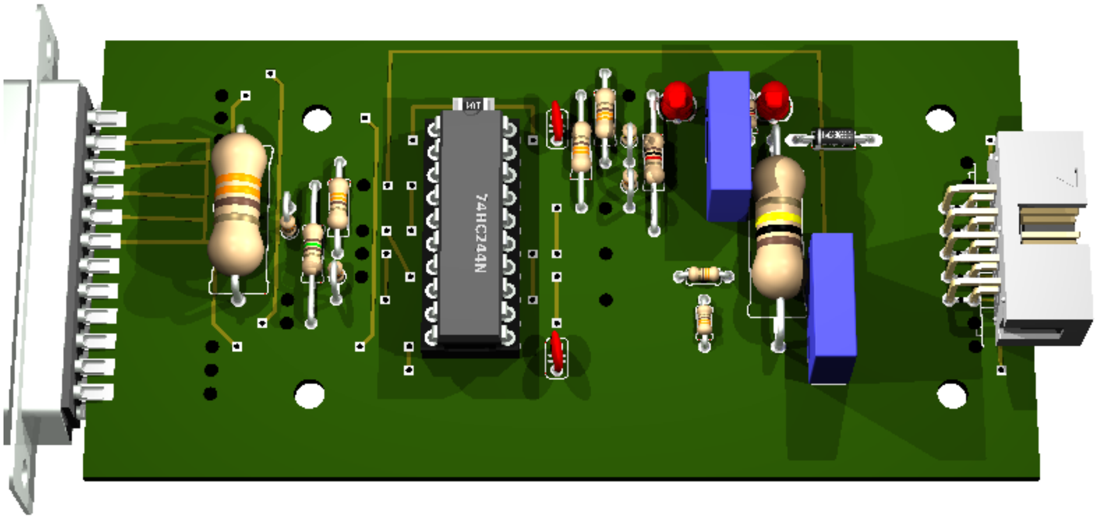
10.5 Partlist

Table 10.1:

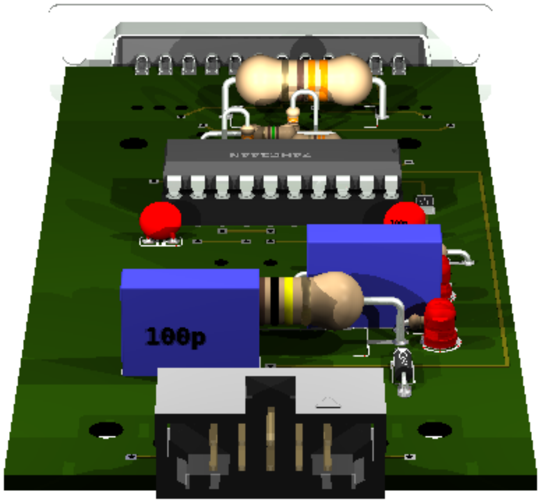
part	value	position
C1	100p	(2.3 0.95)
C3	100n	(2.3 1.95)
C5	100p	(3 1.8)
C6	100p	(3.4 1.1)
D1	1N4148DO35-7	(3.45 1.9)
IC1	74HC244N	(1.95 1.45)
LED1	red	(3.2 2.05)
LED2	red	(2.8 2.05)
R1	330	(1.4 1.6)
R2	150	(1.3 1.4)
R3	330	(1.2 1.55)
R4	1k	(2.7 1.8)
R5	100k	(3.2 1.45)
R6	330	(1.4 1.25)
R7	100k	(2.6 1.85)
R8	330	(1 1.55)
R9	330	(2.9 1.1)
R10	1k	(3.1 2)
R12	330	(2.5 2)
R13	100k	(1.95 2.05)
R14	330	(2.6 1.65)
R15	330	(2.4 1.85)
R16	330	(2.9 1.3)
X1		(0.2 1.4)
X2		(3.95 1.4)

10.6 3D view

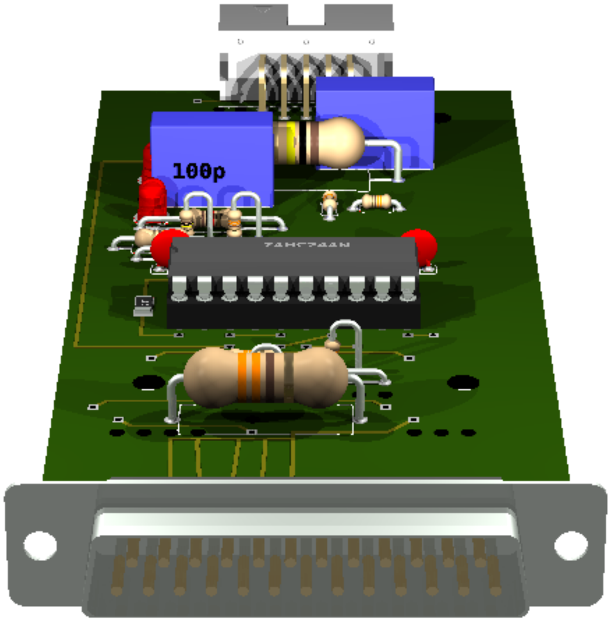
10.6.1 Front



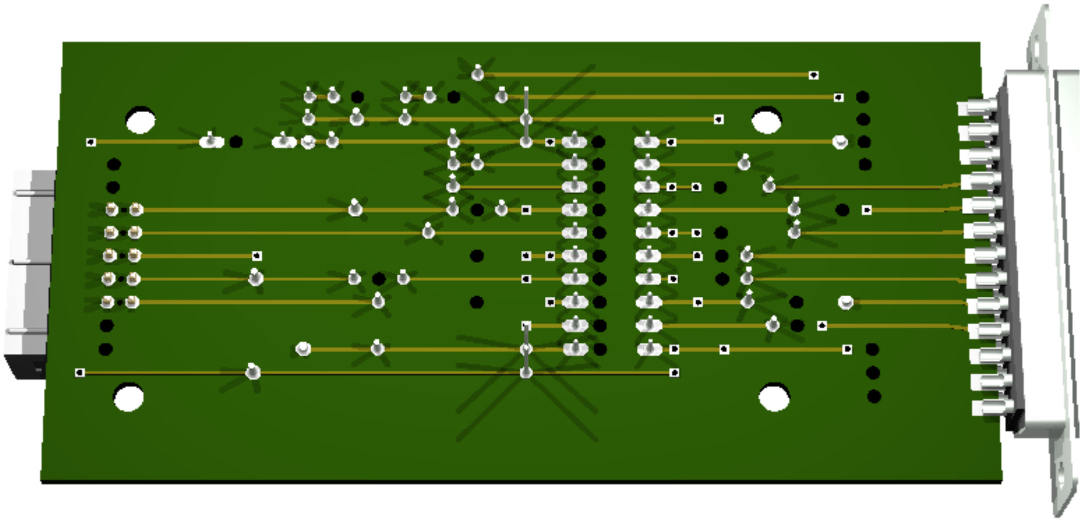
10.6.2 Right side



10.6.3 Left side



10.6.4 Bottom



10.7 Sources

original design

Parallel port specification

ISP pinout

similar designs:

- http://www.sbprojects.com/projects/stk200/

CHAPTER

ELEVEN

USBASP AVR PROGRAMMER

Status: OK

It is used for programming AVR controller and Arduino compatible boards using the USB port.

firmware, design: http://www.fischl.de/usbasp/

USBasp is based on V-USB (http://www.obdev.at/products/vusb/index.html)

11.1 V-USB hardware recommendation

only difference to USBasp: 1.5 kΩ pull-up resistor

http://vusb.wikidot.com/hardware

“Solution B: Level conversion on D+ and D- Level conversion with Zener diodes.

Instead of reducing the AVR’s power supply, we can limit the output voltage on D+ and D- with Zener diodes. We recommend 3.6 V low power types, those that look like 1N4148 (usually 500 mW or less). Low power types are required because they have less capacitance and thus cause less distortion on the data lines. And 3.6 V is better than 3.3 V because 3.3 V diodes yield only ca. 2.7 V in conjunction with an 1.5 kΩ (or more exactly 10 kΩ) pull-up resistor. With 3.3 V diodes, the device may not be detected reliably.

If you use Zener diodes for level conversion, please measure the voltage levels to make sure that the diodes you have chosen match the requirements.

Advantages of the Zener diode approach:

- Low cost.
- Easy to obtain.
- Entire design can be at 5 V.
- AVR can be clocked at high rates.

Disadvantages:

- Not a clean solution, a compromise between all parameters must be found.
- Zener diodes come with a broad range of characteristics, especially at low currents, results may not be reproducible.
- High currents when sending high-level.
- High level is different for signaling and in idle state because signaling uses high currents to drive the diodes while idle state is driven by a 1.5 kΩ pull-up resistor.”

11.2 Makefile

Tested with atmega88. Makefile settings:

```
TARGET=atmega88
HFUSE=0xdd
LFUSE=0xef
```

11.3 Test on Ubuntu

checking:

```
$ lsusb |grep -i 16c0:05dc
Bus 003 Device 006: ID 16c0:05dc VOTI shared ID for use with libusb

$ ls -l /dev/bus/usb/003/006
crw-rw-r-- 1 root root 189, 261 2011-11-05 10:31 /dev/bus/usb/003/006

$ avrdude -patmega88 -cusbasp
avrdude: Warning: cannot query manufacturer for device: error sending control message: Operation not
avrdude: error: could not find USB device "USBasp" with vid=0x16c0 pid=0x5dc
```

The permission should be changed:

```
$sudo nano /etc/udev/rules.d/60-objdev.rules
```

add this line:

```
ATTRS{idVendor}=="16c0", ATTRS{idProduct}=="05dc", GROUP="users", MODE="0666"
```

update rules:

```
$sudo udevadm trigger
```

checking again:

```
$ ls -l /dev/bus/usb/003/006
crw-rw-rw- 1 root users 189, 261 2011-11-05 10:33 /dev/bus/usb/003/006

$ avrdude -patmega88 -cusbasp
avrdude: error: programm enable: target doesn’t answer. 1
avrdude: initialization failed, rc=-1
         Double check connections and try again, or use -F to override
         this check.
avrdude done.  Thank you.
```

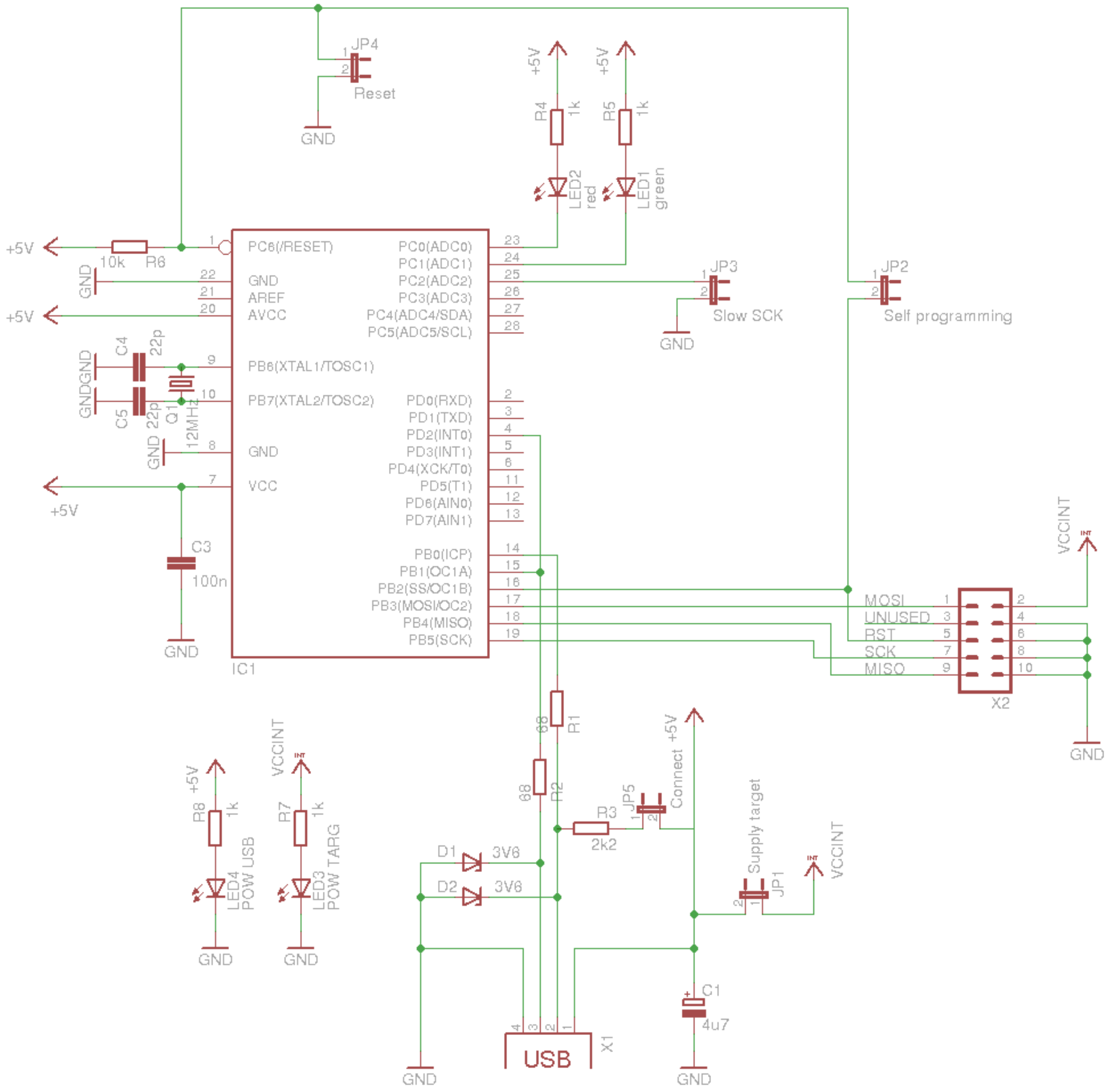
Permission is OK now.

Testing with connected controller:

```
$ avrdude -patmega88 -cusbasp

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.01s

avrdude: Device signature = 0x1e930a
```

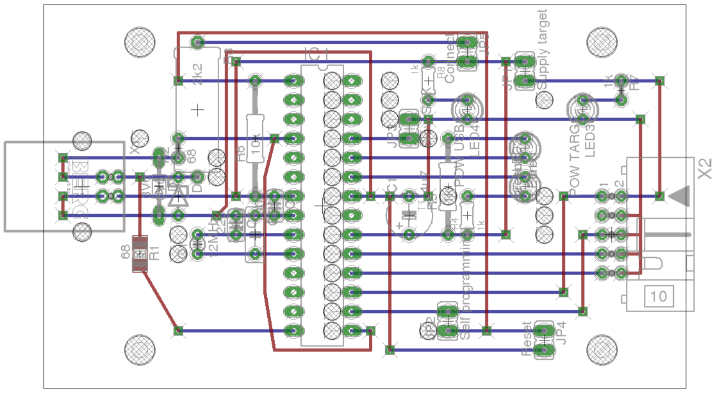
```
avrdude: safemode: Fuses OK

avrdude done.  Thank you.
```

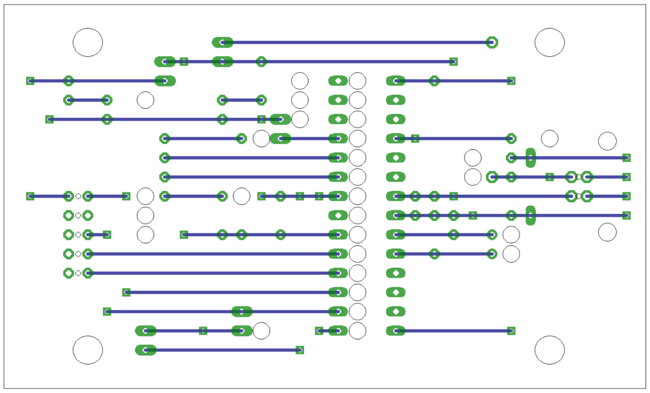
11.4 Schematic

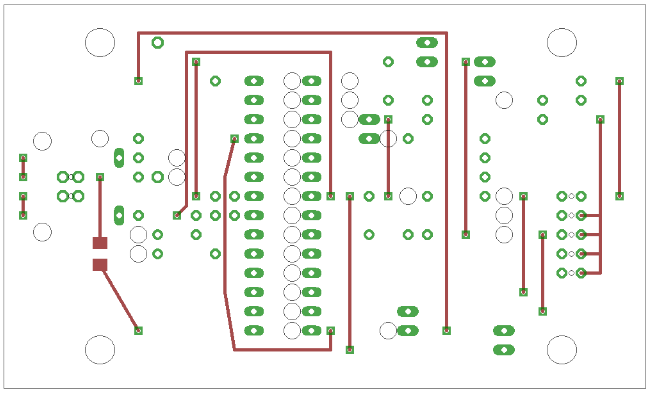


11.5 Board

Normal, bottom mirrored, wires only:


IC1
2k2
68
R1
12MHz
R6
10k
1k
R8
R7
LED3
Supply target
Self programming
Reset
JP4
X2
1
2
10





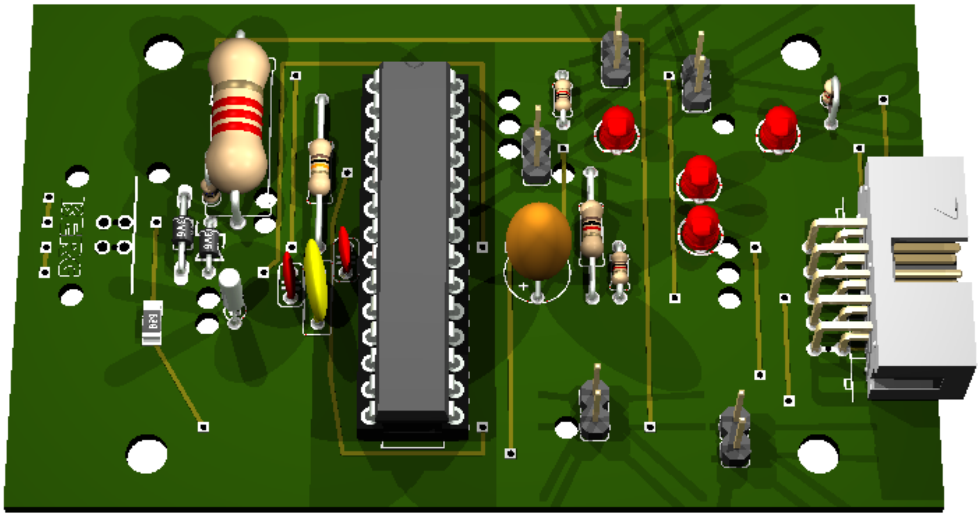
11.6 Partlist

Table 11.1:

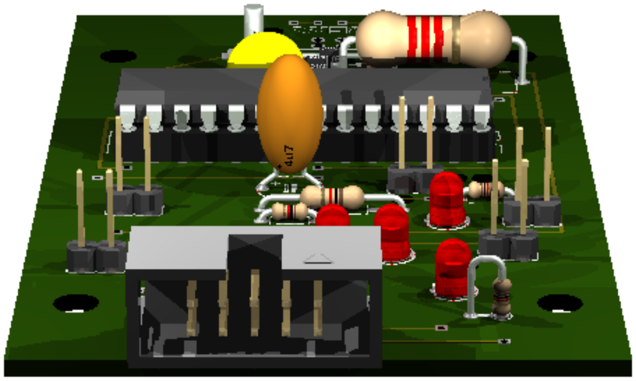
part	value	position
C1	4u7	(2.4 0.8)
C3	100n	(1.7 0.85)
C4	22p	(1.5 0.75)
C5	22p	(1.6 0.7)
D1	3V6	(1.1 0.95)
D2	3V6	(1.2 0.9)
IC1		(1.95 0.85)
JP1	Supply target	(3 1.55)
JP2	Self programming	(2.6 0.25)
JP3	Slow SCK	(2.4 1.25)
JP4	Reset	(3.1 0.15)
JP5	Connect	(2.7 1.65)
LED1	green	(3 1.15)
LED2	red	(3 0.95)
LED3	POW TARG	(3.3 1.35)
LED4	POW USB	(2.7 1.35)
Q1	12MHz	(1.3 0.65)
R1	68	(1 0.6)
R2	68	(1.2 1.15)
R3	2k2	(1.3 1.35)
R4	1k	(2.7 0.8)
R5	1k	(2.6 0.95)
R6	10k	(1.6 1.2)
R7	1k	(3.5 1.45)
R8	1k	(2.5 1.5)
X1		(0.7 0.95)
X2		(3.45 0.7)

11.7 3D view

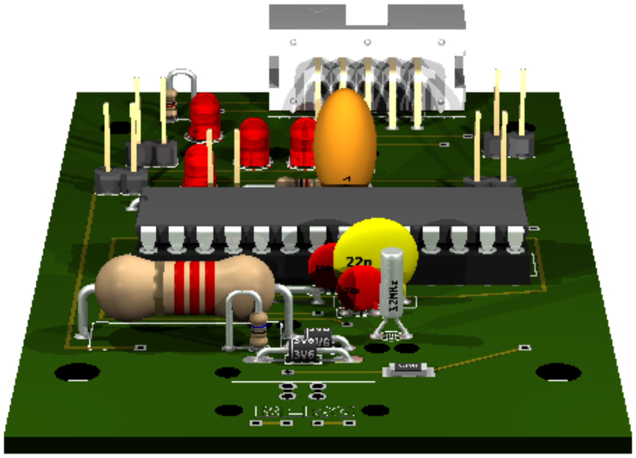
11.7.1 Front



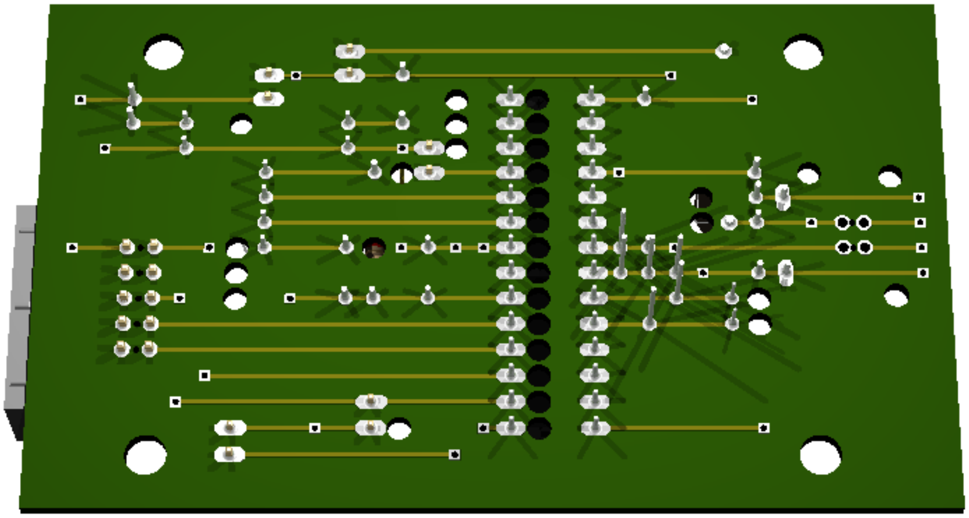
11.7.2 Right side



11.7.3 Left side



11.7.4 Bottom



11.8 Reset

To reset on Ubuntu:

```
#!/usr/bin/env python
import logging
import usb.core
logging.basicConfig(level=logging.DEBUG)
import fcntl

ID_VENDOR = 0x16c0
ID_PRODUCT = 0x05dc
USBDEVFS_RESET = 21780

def find():
    print("searching for device (%x:%x)" % (ID_VENDOR, ID_PRODUCT))
    dev = usb.core.find(idVendor=ID_VENDOR,
                        idProduct=ID_PRODUCT,
                        )
    if not dev:
        print("device not found")
    return dev
```

```
def usbstr(i):
    s=str(i)
    s=’000’[0:3-len(s)]+s
    return s

def usbfs_filename(dev):
    return ’/dev/bus/usb/%s/%s’ % (usbstr(dev.bus), usbstr(dev.address))

def reset1(dev):
    fname=usbfs_filename(dev)
    print("Resetting USB device %s" % fname)
    with open(fname, ’w’) as fd:
        rc = fcntl.ioctl (fd, USBDEVFS_RESET, 0)
        if (rc < 0):
            print("Error in ioctl")
    print("OK")

def reset2(dev):
    dev.reset() # not working

dev=find()
if dev:
    reset1(dev)
```

11.9 Sources

original design

ISP pinout

similar projects:

- http://lategahn.2log.de/index.php?USBASP-Stripboard-layout

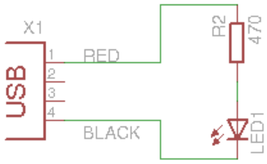
CHAPTER

TWELVE

USB LED

Status: OK

It is used for testing USB power.



X1
USB
1
2
3
4
RED
BLACK
R2
470
LED1